Interspeech 2018
2-6 September 2018, Hyderabad

S4D: Speaker Diarization Toolkit in Python

Pierre-Alexandre Broux[1,2], *Florent Desnous*[2], *Anthony Larcher*[2], *Simon Petitrenaud*[2], *Jean Carrive*[1], *Sylvain Meignier*[2]

[1]French National Audiovisual Institute (INA), Paris, France
[2]Computer Science Laboratory of Le Mans University (LIUM - EA 4023), Le Mans, France

pabroux@ina.fr, florent.desnous@univ-lemans.fr, anthony.larcher@univ-lemans.fr, simon.petit-renaud@univ-lemans.fr, jcarrive@ina.fr, sylvain.meignier@univ-lemans.fr

Abstract

In this paper, we present S4D, a new open-source Python toolkit dedicated to speaker diarization. S4D provides various state-of-the-art components and the possibility to easily develop end-to-end diarization prototype systems. S4D offers a large panel of clustering, segmentation, scoring and visualization algorithms. S4D has been thought to be easily understood, installed, modified and used in order to allow fast transfers of diarization technologies to industry and facilitate development of new approaches. Examples, benchmarks on standard tasks and tutorials are provided in this paper. S4D is an extension of the open-source toolkit for speaker recognition: SIDEKIT.

Index Terms: SIDEKIT, diarization, toolkit, Python, open-source, tutorials

1. Introduction

The diarization task is a necessary pre-processing step for speaker identification [1] or speech transcription [2] when there is more than one speaker in an audio/video recording. For each speaker in a recording, it consists of detecting the time areas where he or she speaks. Each time area, corresponding to a segment, is annotated with an abstract label representing the speaker. Thus, the diarization task allows to determine *who spoke when*. This domain is still an active research area since there are many unsolved problems such as detection of overlapped speech [3] or labeling of speech overlapping with music.

For the diarization task, few toolkits are available. Most of them are dedicated to research. Quick transfers of new technologies to industry require tools which are close to industrial standards. So as to reach this purpose, a diarization toolkit should comply with some requirements:

- be easy to understand, modify, install and use;
- enable end-to-end diarization system development;
- offer various state-of-the-art algorithms;
- manage standard data formats to allow compatibility with other tools.

To address the lacks of existing toolkits, we developed S4D, a new toolkit for diarization fulfilling the mentioned requirements and facilitating the development of new approaches.

In this paper, we first present the context in which S4D has been developed. We give then a detailed description of S4D contents before providing a guide to develop a broadcast news diarization system. Finally, we explain how to deploy S4D before offering a few perspectives.

2. Context

This section presents the context in which S4D has been developed, other existing tools and the link with SIDEKIT [4].

2.1. Comparison and compatibilities with existing tools

Few tools are freely available for speaker diarization. S4D has been designed to overcome limitations of those tools.

LIUMSpkDiarization [5, 6] is a toolkit for diarization written in Java. It includes most state-of-the-art methods in the diarization field. This toolkit was developed by the Computer Science Laboratory of Le Mans University (LIUM) for French ESTER2 evaluation campaign [7], where it obtained the best results for the task of diarization of broadcast news in 2008. This toolkit has two main drawbacks: it is no longer being updated and it can only be executed via command lines thanks to a jar file.

Pyannote.metrics [8] is a toolkit for reproducible evaluation, diagnostic and error analysis of diarization systems. It is a regularly updated project with a wide selection of metrics. S4D includes certain metrics from this toolkit to offer greater ease of use.

Pyannote.audio [9] is a toolkit for diarization. It only proposes state-of-the-art methods developed by using the oriented object paradigm in which it is easy to extend. Moreover, it requires a considerable learning time.

2.2. SIDEKIT and S4D

SIDEKIT is an open source package for speaker and language recognition developed by Anthony Larcher, Kong Aik Lee and Sylvain Meignier [4] which provides an end-to-end toolchain including various state-of-the-art algorithms. SIDEKIT for Diarization (S4D) is an open source package extension of SIDEKIT dedicated to diarization. The aim of S4D is to provide an educational and efficient toolkit for diarization encompassing the whole chain of treatment that goes from the audio data to the analysis of the system performance. Furthermore, both SIDEKIT and S4D have completely been written in Python and tested on several platforms under Python 3 for both Linux and MacOS.

3. What is in S4D?

This section describes several uses currently offered by S4D.

3.1. Segmentation

The segmentation detects the instantaneous change points corresponding to segment boundaries. The proposed algorithm is based on the detection of local maxima. It detects the change points through a Gaussian Divergence (GD) [10], computed using Gaussians. The left and right Gaussians are estimated over a window sliding along the whole signal. A change point, i.e. a segment boundary, is present in the middle of the window when the Gaussian divergence score reaches a local maximum.

After a GD segmentation, a second pass over the signal fuses consecutive segments of the same speaker from the start to the end of the recording. The employed measure for the fusing is the ΔBIC [11] based on Bayesian Information Criterion. Alternatively, it is possible to use the BIC Square Root distance for the value of the penalty factor in the ΔBIC, as defined in [12].

3.2. Clustering

In order to group clusters, S4D offers a certain number of methods.

3.2.1. HAC BIC

The algorithm is based upon a Hierarchical Agglomerative Clustering (HAC). Each cluster is modeled by a Gaussian. The ΔBIC measure [11] is employed to select the candidate clusters to be grouped as well as to stop the merging process. The two closest clusters i and j are merged at each iteration until $\Delta BIC_{i,j} > 0$.

3.2.2. HAC CLR

The HAC CLR merges a set of clusters thanks to a HAC algorithm. The CLR (Cross Likelihood Ratio) score [13] is used as the dissimilarity measure as well as the stop criterion. This score requires the Universal Background Model (UBM) for the computation and to eventually adjust used data models with the MAP algorithm [14]. The lowest CLR score allows to select the two clusters to merge at each iteration. The merging process stops when the score exceeds a threshold set a priori.

3.2.3. ILP IV

The Integer Linear Programming I-Vector (ILP IV) clustering [15] extracts an i-vector for each cluster and computes the distances among all of them (PLDA [16], cosine [17] or Mahalanobis [18]). ILP clustering was inspired by the k-medoids algorithm which choose k observations as class centers. For the ILP IV, this number k is determined automatically. We look for K centers which cover all the i-vectors such as each one is assigned to only one center and has a distance of less than δ from its center. This problem is solved using the GNU Linear Programming Kit (GLPK) package which is intended for solving large-scale Linear Programming (LP).

So as to save execution time, a search of connected components (CC) can be done [19]. The distances below δ represent connected components with clusters as nodes and distances as edges. The ILP IV clustering is then applied for each connected component which is not in a form of a star graph. A star is just one or several nodes only connected to a same node.

3.2.4. HAC IV

This clustering process is based upon a HAC algorithm. Each cluster is modeled by an i-vector and the distances among all of them are computed thanks to the PLDA, cosine or Mahalanobis score. This distance is the measure employed to select the clusters to be grouped as well as to stop the clustering process. The two clusters with the shortest distance are merged at each iteration until the distance reaches a threshold set a priori. The HAC linkage rests upon a method from the SciPy toolkit [20]. Thus all the merging rules (i.e. single or complete linkage) are available.

3.3. HMM/Viterbi

In order to adjust the segment boundaries or to detect speech/non-speech regions, a Viterbi decoding is frequently used in the speech field. Each state of the Hidden Markov Model (HMM) represents a speaker or a data class (speech, silence, noise, etc.) modeled by a Gaussian Mixture Model (GMM). A GMM is learned thanks to the EM-ML algorithm [21] over the segments of the cluster. GMM are preferentially used instead of single Gaussians since the available data for speakers are important. The insertion penalty of a cluster during a decoding is fixed experimentally.

3.4. Evaluation and visualization

S4D offers the possibility to compute various metrics such as the DER, the Speech/Non-speech evaluation and a measure of correction cost: the HCIQ. New specific ways to visualize a diarization are proposed as well.

3.4.1. DER

The Diarization Error Rate (DER) is the well-known metric used to measure the quality of a diarization [22]. DER was introduced by the NIST (National Institute of Standards and Technology) as the fraction of speaking time which is not attributed to the correct speaker, using the best matching between references and hypothesis speaker labels. The DER measure is the sum of three errors:

$$DER = \frac{M + S + F}{T}, \tag{1}$$

where M, F and S respectively correspond to the cumulated duration of the missed speakers, the false alarms and the confusion among speakers and T is the total speech duration.

To compute this measure, S4D uses the evaluation tool developed by the LNE [23] like Pyannote.metrics [8].The LNE evaluation tool relies on the Hungarian algorithm [24], which gives an optimal solution for the problem of assignment in a polynomial time allowing to evaluate cross-show diarization with a large number of reference and hypothesis speakers.

3.4.2. Speech/Non-speech evaluation

The Speech/Non-speech evaluation (SNS) measures the detection of speech and non-speech frames in a diarization. SNS is the fraction of well-detected speaking and non-speaking time, using the best matching between reference and hypothesis speaker labels. The SNS measure is computed as follows:

$$SNS = \frac{S + N}{T}, \tag{2}$$

where S is the well-detected speech duration, N the well-detected non-speech duration and T the total speech duration.

3.4.3. HCIQ: a measure of the correction cost

Since the DER measures the diarization quality in terms of duration, it is not relevant to evaluate the human effort to correct or to produce a speaker diarization. We have introduced a new

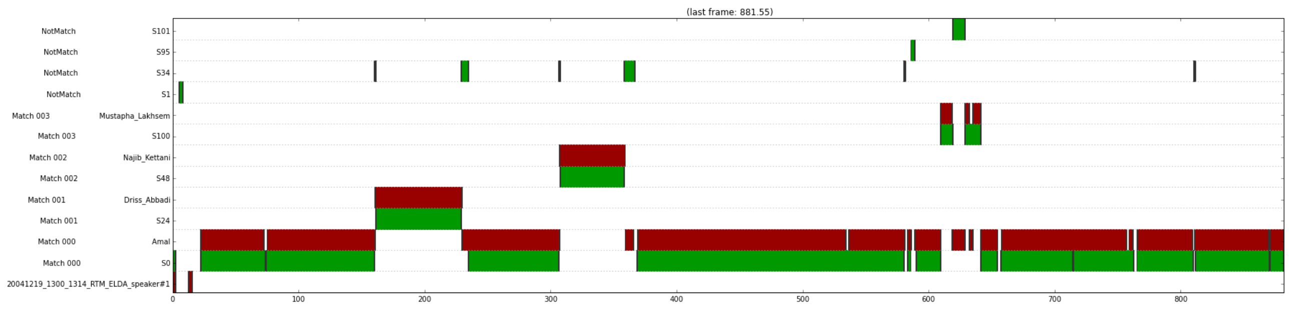

Figure 1: *Example of a DER visualization in S4D*

metric which we called Human-Computer Interaction Quantity (HCIQ) [25]. The HCIQ can be computed both for assisted systems and systems where a human corrects the diarization alone. By the way, this measure allows to compare different assisted diarization systems in a objective way. As well as the DER and SNS measures, HCIQ can be computed for a recording or for a set of recordings:

$$HCIQ = \frac{\sum_{i=1}^{K} w_i n_i}{d}, \qquad (3)$$

where d the duration of the corrected and validated annotations, i the index for a correction action type, w_i the associated cost, n_i the number of times an annotator has applied this action type and K the number of different action types. When HCIQ increases, the amount of corrections increases also. The available action types depend of the chosen annotation software but the minimum required actions to entirely correct a diarization are "*Create a boundary*", "*Delete a boundary*", "*Create a speaker label*" and "*Change the speaker label*". These minimum required actions are more precisely described in [25].

In order to estimate the diarization corrections and to facilitate the HCIQ evaluation, S4D offers some algorithms simulating a human annotator.

3.4.4. *Diarization visualization*

S4D offers two visualizations. One of them is dedicated to the DER (figure 1) and the other one to the clustering process (figure 2). The former one allows to easily visualize the correspondence among the hypothesis and the reference clusters thanks to the DER.

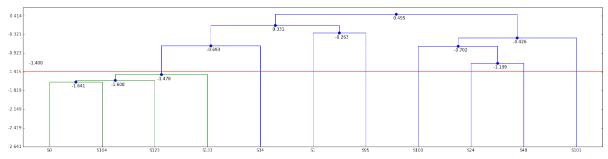

Figure 2: *Example of a dendrogram visualization in S4D*

The latter one illustrates a dendrogram so as to find easily the order of the clustering. Unfortunately, these visualizations provided in Python offer less flexibility than the ones provided by annotation softwares such as *ELAN* [26].

3.5. Diarization format

S4D uses several file formats to read or write a diarization. However, under Python, there is only one representation format.

3.5.1. *File format*

Several output formats already exist in literature or can be used to represent a diarization. The three main formats are the following: the NIST format, the MDTM (Meta Data Time Marks) format and the RTTM (Rich Transcription Time Markers) format.

Even though S4D can read and write any one of these formats, another one, named the LIUM format, is similar but more adapted to S4D. It is developed by the LIUM laboratory and is inspired by the MDTM and the RTTM format for a diarization purpose. Each line of the LIUM format file represents a segment and follows this structure:

$$\textit{show start duration gender channel env speaker}, \qquad (4)$$

where $show$ is the recording name, $start$ is the segment start, $duration$ is the segment length, $gender$ is the speaker gender, $channel$ is the band type (for instance: telephone or studio), env is the environment type (for instance: music or speech only) and $speaker$ is the speaker label. The LIUM format is used by LIUMSpkDiarization [5] as well.

3.5.2. *Python format*

Each row of a diarization file in S4D is a segment described by n entities identified by an attribute name. The default segment is composed of 5 attributes respecting this following order:

$$\textit{show cluster type start stop}, \qquad (5)$$

where $show$ is the recording name, $cluster$ is the segment cluster (i.e. the speaker name), $type$ is the cluster type (speaker or head), $start$ is the segment starting time (in centiseconds) and $stop$ is the segment end time. Attributes can by added or removed but only the attributes matching with the available file formats (see 3.5.1) can be written in a output file. The other attributes are not written. Figure 3 shows an example of a diarization in S4D with default segments. A segment is a mere

```
[
  attribut definition  : ['show', 'cluster', 'cluster_type', 'start', 'stop']
  row 0: ['BFMTV_BFMStory_2012-01-10_175800', 'S1', 'speaker', 0, 100]
  row 1: ['BFMTV_BFMStory_2012-01-10_175800', 'S2', 'speaker', 100, 200]
  row 2: ['BFMTV_BFMStory_2012-01-10_175800', 'S1', 'speaker', 300, 400]
  row 3: ['BFMTV_BFMStory_2012-01-10_175800', 'S18', 'speaker', 350, 450]
  row 4: ['BFMTV_BFMStory_2012-01-10_175800', 'S2', 'speaker', 310, 320]
  row 5: ['BFMTV_BFMStory_2012-01-10_175800', 'S3', 'speaker', 470, 500]
]
```

Figure 3: *Example of a diarization in S4D*

portion of a show with a cluster label as an annotation. With the frame rate as the unit, it defines a kind of slice from $start$ until $stop - 1$. A diarization could draw data from several shows which is very useful in a batch mode context (for instance: a cross-show diarization or computing log likelihood ratio).

4. How to develop a broadcast news diarization system

A state-of-the-art diarization system for broadcast news is typically divided in three steps (figure 4).

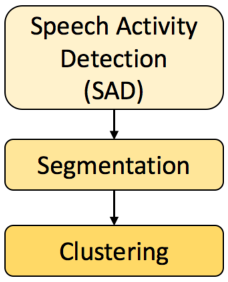

Figure 4: *Typical steps of a state-of-the-art diarization system for broadcast news*

For the SAD step, we use a GMM-HMM model trained with SIDEKIT. For the segmentation and the clustering steps, we use the algorithms described in the subsections 3.1, 3.2 and 3.3.

The Gaussian Divergence algorithm is firstly used to perform the segmentation step. This algorithm uses Gaussians with diagonal covariance matrices. The left and right Gaussians are estimated over a five-second window (2.5 seconds for each Gaussian) sliding along the whole signal. Then ΔBIC is used (with a threshold $= 2$) to fuse consecutive segments of the same speaker.

For the clustering step, first we use a hierarchical clustering (HAC) using the ΔBIC measure (threshold $= 3$) and Gaussians with a full covariance matrix as described in the subsection 3.2.1. The segment boundaries are then readjusted using a Viterbi decoding as described in the subsection 3.3 (threshold $= -250$). The final part of the clustering step involves i-vectors extracted using a model trained on the ESTER1 training corpus. Once the distance matrix is computed using PLDA, it is processed as described in sub-subsection 3.2.3. The threshold varies from -100 to 100 with a step of 10. The results are then evaluated (see 3.4) to select the best threshold ([27] is an example of the use of S4D with this state-of-the-art approach).

Table 1: *Evaluations for various corpora*

Corpus	**SNS (%)**	**DER (%)**	**HCIQ**
ESTER test 2003	0.93	6.48	0.82
ESTER test 2009	1.29	6.23	1.15
ETAPE test 2012	3.74	15.56	1.94
REPERE test 2013	1.46	9.24	1.91

Table 2: *Execution time in minutes for various corpora. GS: GD with the second pass; HB: HAC BIC; HV: HMM/Viterbi; II: ILP IV*

Corpus	**GS**	**HB**	**HV**	**II**
ESTER test 2003	0.21	0.64	3.68	12.75
ESTER test 2009	0.14	0.33	2.13	9.31
ETAPE test 2012	0.17	0.84	2.94	12.22
REPERE test 2013	0.29	1.28	5.35	18.57

Table 1 illustrates the best results for this system. In this table, the HCIQ measure is computed by using the actions of *Transcriber* [28]. These actions are the same as the minimum required action in sub-subsection 3.4.3. The associated action times are available in [25]. Table 2 shows the execution times for each step of the diarization process. The results are given for 4 French broadcast news corpora used during past evaluation campaigns [7, 29, 30, 31].

5. Deploying S4D

This section describes the S4D installation and the available tools for the community.

5.1. Installation

S4D is available via the PyPI repository with the command line "*pip install s4d*" which will install all required Python packages at once. S4D sources can also be obtained by cloning the S4D GIT repository with the command line "*git clone https://git-lium.univ-lemans.fr/Meignier/s4d.git*".

5.2. Tools for the community

Firstly S4D has been released under LGPL License to allow a wider usage of the code that, hopefully, could be beneficial to the community. The structure of the core package makes use of a limited number of classes in order to facilitate the readability and reusability of the code. Secondly a web portal for supporting the use of S4D is available at *http://www-lium.univ-lemans.fr/sidekit/s4d/*. This portal includes tutorials, a complete documentation, links on related tools as well as references on related articles. Moreover a mailing list for developers and users has been set up to exchange remarks and help. The registration of this mailing list is also available via the web portal. Finally the contributions for the GIT repository will be welcome.

6. Discussion

We have introduced S4D, a new open-source toolkit for the diarization task. It is a comprehensive toolkit offering an end-to-end tool-chain with various ready-to-use state-of-the-art algorithms. S4D allows to easily develop systems for broadcast news but also for other tasks (meeting, telephone conversations). It is very useful to create offline diarization system but is not adapted yet for online diarization system or treatments in stream. The resulting diarization system is nonetheless time efficient, as it processes the total 40 hours of our test corpus in 70 minutes (see Table 2), which corresponds to less than 3% of the total audio duration. This toolkit is maintained for an indefinite period. It will implement new methods and metrics according to speaker diarization advances. In the near future, Artificial Neural Network (ANN) [32] and Binary Key (BK) [33] methods for segmentation and clustering will be implemented.

7. References

[1] J.-F. Bonastre, P. Delacourt, C. Fredouille, T. Merlin, and C. Wellekens, “A speaker tracking system based on speaker turn detection for NIST evaluation,” in *Acoustics, Speech, and Signal Processing, 2000. ICASSP’00. Proceedings. 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. II1177–II1180.

[2] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, “Speaker diarization: A review of recent research,” *ieee-tsap*, vol. 20, no. 2, pp. 356–370, Feb 2012.

[3] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland, “Overlapped speech detection for improved speaker diarization in multiparty meetings,” in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 4353–4356.

[4] A. Larcher, K. A. Lee, and S. Meignier, “An extensible speaker identification SIDEKIT in python,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5095–5099.

[5] S. Meignier and T. Merlin, “LIUM SpkDiarization: An open source toolkit for diarization,” in *CMU SPUD Workshop*, 2010.

[6] M. Rouvier, G. Dupuy, P. Gay, E. Khoury, T. Merlin, and S. Meignier, “An open-source state-of-the-art toolbox for broadcast news diarization,” in *Interspeech*, 2013.

[7] S. Galliano, G. Gravier, and L. Chaubard, “The ESTER 2 evaluation campaign for the rich transcription of french radio broadcasts,” in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[8] H. Bredin, “pyannote. metrics: A toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems,” in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, 2017.

[9] ——, “pyannote.audio,” https://github.com/pyannote/pyannote-audio, 2017.

[10] C. Barras, X. Zhu, S. Meignier, and J.-L. Gauvain, “Multistage speaker diarization of broadcast news,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1505–1512, 2006.

[11] G. Schwarz *et al.*, “Estimating the dimension of a model,” *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[12] T. Stafylakis, V. Katsouros, and G. Carayannis, “The segmental Bayesian information criterion and its applications to speaker diarization,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 857–866, 2010.

[13] D. A. Reynolds, E. Singer, B. A. Carlson, G. C. O’Leary, J. J. McLaughlin, and M. A. Zissman, “Blind clustering of speech utterances based on speaker and language characteristics,” in *Fifth International Conference on Spoken Language Processing*, 1998.

[14] J.-L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE transactions on speech and audio processing*, vol. 2, no. 2, pp. 291–298, 1994.

[15] M. Rouvier and S. Meignier, “A global optimization framework for speaker diarization,” in *Odyssey 2012*, 2012.

[16] Y. Jiang, K. A. Lee, Z. Tang, B. Ma, A. Larcher, and H. Li, “PLDA modeling in i-vector and supervector space for speaker verification,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[17] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[18] P.-M. Bousquet, D. Matrouf, and J.-F. Bonastre, “Intersession compensation and scoring methods in the i-vectors space for speaker recognition,” in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[19] G. Dupuy, S. Meignier, and Y. Esteve, “Is incremental cross-show speaker diarization efficient for processing large volumes of data?” in *Interspeech*, 2014.

[20] E. Jones, T. Oliphant, and P. Peterson, “{SciPy}: Open source scientific tools for {Python},” 2014.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[22] NIST, “The Rich Transcription Spring 2003 (RT-03S) evaluation plan,” February 2003.

[23] O. Galibert, “Methodologies for the evaluation of speaker diarization and automatic speech recognition in the presence of overlapping speech.” in *INTERSPEECH*, 2013, pp. 1131–1134.

[24] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval Research Logistics (NRL)*, vol. 2, no. 1-2, pp. 83–97, 1955.

[25] P.-A. Broux, D. Doukhan, S. Petitrenaud, S. Meignier, and J. Carrive, “Computer-assisted speaker diarization: How to evaluate human corrections,” in *LREC 2018, 11th edition of the Language Resources and Evaluation Conference*, 2018.

[26] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes, “ELAN: A professional framework for multimodality research,” in *Proceedings of LREC*, vol. 2006, 2006, p. 5th.

[27] R. Yin, H. Bredin, and C. Barras, “Neural speech turn segmentation and affinity propagation for speaker diarization,” in *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association*, 2018.

[28] C. Barras, E. Geoffrois, Z. Wu, and M. Liberman, “Transcriber: Development and use of a tool for assisting speech corpora production,” *Speech Communication*, vol. 33, no. 1, pp. 5–22, 2001.

[29] G. Gravier, J.-F. Bonastre, E. Geoffrois, S. Galliano, K. McTait, and K. Choukri, “The ESTER evaluation campaign for the rich transcription of french broadcast news.” in *LREC*, 2004.

[30] O. Galibert, J. Leixa, G. Adda, K. Choukri, and G. Gravier, “The ETAPE speech processing evaluation.” in *LREC*. Citeseer, 2014, pp. 3995–3999.

[31] J. Kahn, O. Galibert, L. Quintard, M. Carré, A. Giraudel, and P. Joly, “A presentation of the REPERE challenge,” in *Content-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on*. IEEE, 2012, pp. 1–6.

[32] R. Yin, H. Bredin, and C. Barras, “Speaker change detection in broadcast TV using bidirectional long short-term memory networks,” in *Proc. Interspeech 2017*, 2017, pp. 3827–3831.

[33] X. Anguera and J.-F. Bonastre, “Fast speaker diarization based on binary keys,” in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4428–4431.