

Project ANR-15-CE39-0010-01

Online Diarization Enhanced by recent Speaker identification and Structured prediction Approaches

ODESSA

Final scientific report

1 Summary

ODESSA focuses on the detection and interception of cyber-crime and terrorism involving voice over Internet protocol (VoIP) communications. VoIP is used to describe the transmission technology for delivering voice communications using packet-switched networks and related protocols. A plethora of different VoIP software applications are now available, and a good number of them are free, e.g. Skype, Google+ Hangouts, iCall and Viber. Such tools can be used to spread terrorist ideologies, to recruit new members and to prepare, plan and coordinate terrorist attacks. The ability to detect and intercept such communications is thus an opportunity to stem the rise of cyber-criminality. The goal of the project is to provide an online speaker diarization system that help detect and recognize the speakers of interest.

Therefore, in this project³, Idiap has contributed to several components of such online diarization system:

- Collected the first database of multi-party speech data over Internet telephony, soon to be publicly available. We have used GotoMeeting VoIP recording application to record online communications between 14 pairs of people. We have also annotated the speech in each conversation. The database and the annotations will be publicly available for download and a database interface for the use with *pyannotate.audio* speech diarization framework [1] is already available¹.
- Contributed to the state of the art *pyannotate.audio* framework² for speech diarization. Based on PyTorch machine learning framework, this framework provides a set of trainable end-to-end neural building blocks that can be combined and jointly optimized to build speaker diarization pipelines.
- Evaluated state of the art diarization systems in cross-database scenario to understand how well such systems perform under ‘unseen’ testing conditions. In a realistic scenario, a diarization system is typically trained and fine tuned on one set of data but is deployed in another environment with different test data. We conducted such tests by training and evaluating such system across several public databases.

¹<https://github.com/pyannotate/pyannotate-db-odessa-ip>

²<https://github.com/pyannotate/pyannotate-audio>

- Contributed to the low-latency speaker spotting problem. Related to security and intelligence applications, the task involves the spotting, as soon as possible, of known speakers in multi-speaker audio streams. This problem is an extension of the diarization and automated speaker verification (ASV) problems with an additional restriction on the amount of time that the system has for a decision about the current speaker’s identity.
- The datasets, experiments, source code, and the documentation of this project are made publicly available³ to facilitate the research in this area further.

³<https://gitlab.idiap.ch/odessa/bob.pyannote>

2 Database

The data collection for ODESSA VoIP database was carried out using Idiap’s GotoMeeting recording application. The devices used in the meetings are PCs. All the meeting material is stored in MPEG4 and WAV audio formats. Only voices of the speakers were recorded. The duration of each conversation is 2 minutes maximum.

The database contains 42 conversations in English between 2 speakers. The total number of speakers is 14. All participants have signed consent form by agreeing for the collected data to be used for research purposes. The conversation scenario includes the two speakers reading scripted lines from the script prepared in advance. There is no cross talk between speakers. Each of the speakers used a PC to connect. Each recording session is a brief transcribed VoIP conversation between two speakers. The session manager used third PC to record the session while muting himself.

All audio files are manually annotated by Idiap and the ground truth is stored in Text and RTTM formats. The annotations include the beginnings and ends of the speech for each speaker with local and global speaker IDs. Each transcribed reference is associated with its corresponding session, so that the database could also be used for speech diarization, speech recognition, speaker recognition, and low-latency speaker spotting tasks.

3 Diarization framework

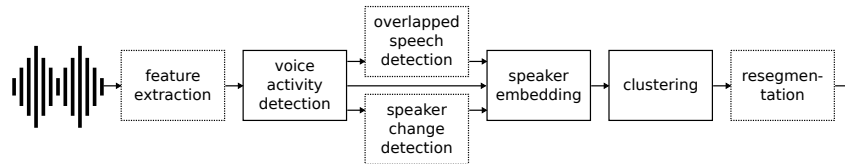


Figure 1: `pyannote.audio` provides a collection of modules that can be jointly optimized to build a speaker diarization pipeline.

Speaker diarization is the task of partitioning an audio stream into homogeneous temporal segments according to the identity of the speaker. As depicted in Figure 1, this is usually addressed by putting together a collection of building blocks, each tackling a specific task (e.g. voice activity detection, clustering, or re-segmentation).

`pyannote.audio`² provides a unified framework to train (usually recurrent) neural networks for several speaker diarization sub-modules, including voice activity detection [2], speaker change detection [1], overlapped speech detection [3], and even re-segmentation [4].

Because processing long audio files of variable lengths is neither practical nor efficient, `pyannote.`

`audio` relies on shorter fixed-length sub-sequences. At training time, fixed-length sub-sequences are drawn randomly from the training set to form mini-batches, increasing training samples variability (data augmentation) and training time (shorter sequences). At test time, audio files are processed using overlapping sliding windows of the same length as used in training. For each time step t ,

this results in several overlapping sequences of K -dimensional prediction scores, which are averaged to obtain the final score of each class.

`pyannote.audio` provides a collection of command line tools for training, validation, and application of modules listed in Figure 1. Reproducible research is facilitated by the systematic use of `pyannote.metrics` [5] and configuration files, while strict enforcement of train/dev/eval split with `pyannote.database` ensures machine learning good practices.

It also comes with a collection of pre-trained models for voice activity detection, speaker change detection, overlapped speech detection, and speaker embedding. While speaker embeddings were trained and tested on VoxCeleb [6], all other models (including the full diarization pipeline) were trained, tuned, and tested on three different datasets, covering a wide range of domains: meetings for AMI [7], broadcast news for ETAPE [8], and up to 11 different domains for DIHARD [9].

4 Assessment of cross database diarization

The goal of this study was to study³ the effect of using different publicly available datasets for training different parts of diarization pipeline. We have chosen several popular databases for training, including AMI dataset [7], CallHomeSRE⁴ a NIST SRE 2000 CallHome subset (the R65-8-1 folder), CallHome⁵ of CABank corpora, REPERE corpus [10], ESTER corpus [11], LibriSpeech [12], and Odessa⁶. An open source toolkit *pyannote.audio*² was used throughout the training and evaluation.

The following data collections were used for training voice activity detection (VAD), speaker change detection (SCD), and speaker embeddings (EMB), as well as, to select best performing models on validation sets and selection of hyper-parameters:

1. CallHomeSRE, subset of NIST SRE 2000 of different language speakers, which was used as a standalone database for training.
2. CallHome English subset from CABank corpora, which was used for training, validation (development subset), and the selection of hyper-parameters (development subset).
3. SmallMeta — a smaller collection of databases, including CallHomeSRE (full database), CallHome (train subset), LibriSpeech (other-train subset), and AMI (train subset). This collection was used for training different models of diarization pipeline.
4. LargeMeta — a larger collection of databases, including SmallMeta plus REPERE (train subsets of phase 1 and 2), ESTER (train subsets of version 1 and 2), and LibriSpeech (clean-train subset). This collection was used for training different models of diarization pipeline.
5. AMI corpus, which was used for models validation (development subset) and the selection of hyper-parameters (development subset).

⁴<https://catalog.ldc.upenn.edu/LDC2001S97>

⁵<https://ca.talkbank.org/browser/index.php?url=CallHome/eng/>

⁶<https://github.com/pyannote/pyannote-db-odessa-ip>

VAD, SCD, and EMB models were trained using *pyannote.audio* on the above databases for 1000 epochs and, for each model, the best epochs were selected based on the validation using either CallHome or AMI databases. The hyper-parameters for the diarization were selected using the Ruiqing Yin *et al.* [4] approach on development sets of either CallHome or AMI databases, i.e., the same sets that were use in validation.

4.1 Cross-database evaluation results

Evaluation of speaker diarization pipelines in terms of diarization error rate are shown in Table 1.

Table 1: Evaluation of speaker diarization system in terms of diarization error rate (%) in cross-database scenario.

Evaluated on	Subset	Models trained on	Validated on	DER (%)
AMI	development	CallHome	AMI	53.21
AMI	development	CallHomeSRE	AMI	54.44
AMI	development	SmallMeta	AMI	49.50
AMI	development	LargeMeta	AMI	47.04
AMI	development	CallHome	CallHome	62.51
AMI	development	CallHomeSRE	CallHome	59.52
AMI	development	SmallMeta	CallHome	66.20
AMI	development	LargeMeta	CallHome	69.34
AMI	test	CallHome	AMI	52.66
AMI	test	CallHomeSRE	AMI	50.08
AMI	test	SmallMeta	AMI	47.01
AMI	test	LargeMeta	AMI	45.09
AMI	test	CallHome	CallHome	58.45
AMI	test	CallHomeSRE	CallHome	55.64
AMI	test	SmallMeta	CallHome	62.66
AMI	test	LargeMeta	CallHome	67.74
CallHome	development	CallHome	AMI	63.25
CallHome	development	CallHomeSRE	AMI	58.12
CallHome	development	SmallMeta	AMI	51.24
CallHome	development	LargeMeta	AMI	58.12
CallHome	development	CallHome	CallHome	36.85
CallHome	development	CallHomeSRE	CallHome	39.68
CallHome	development	SmallMeta	CallHome	39.11
CallHome	development	LargeMeta	CallHome	33.07
CallHome	test	CallHome	AMI	57.81
CallHome	test	CallHomeSRE	AMI	59.13
CallHome	test	SmallMeta	AMI	55.98
CallHome	test	LargeMeta	AMI	62.16
CallHome	test	CallHome	CallHome	43.79
CallHome	test	CallHomeSRE	CallHome	45.25
CallHome	test	SmallMeta	CallHome	47.49
CallHome	test	LargeMeta	CallHome	44.79
ODESSA	test	CallHome	AMI	65.69
ODESSA	test	CallHomeSRE	AMI	58.97
ODESSA	test	SmallMeta	AMI	80.54
ODESSA	test	LargeMeta	AMI	78.87
ODESSA	test	CallHome	CallHome	63.65
ODESSA	test	CallHomeSRE	CallHome	66.73
ODESSA	test	SmallMeta	CallHome	80.38
ODESSA	test	LargeMeta	CallHome	79.86

Since ODESSA database is small in size, the whole database is treated as

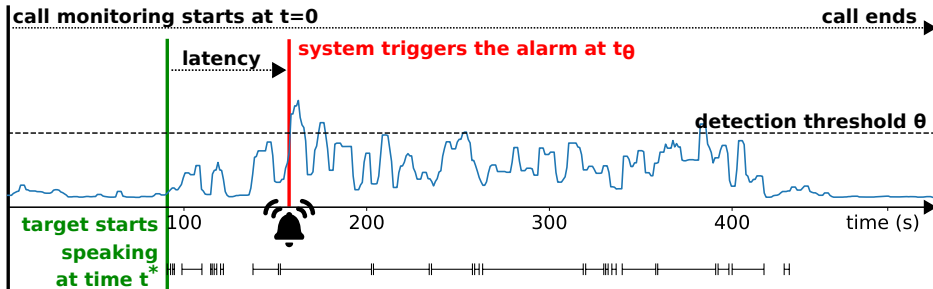


Figure 2: Low-latency speaker spotting systems aim at detecting a target speaker with the lowest possible latency.

one *Test* set. Therefore, in all experiments with ODESSA database, the models trained on other databases were used. Additional results of diarization system trained on AMI and VoxCeleb databases and evaluated on ODESSA database are shown in Table 2. From this table, we can note the importance of which database is used for tuning the hyper-parameters of the diarization system, as the DER values are significantly smaller when the parameters are tuned and tested on the same ODESSA database.

Table 2: Evaluation of speaker diarization pipelines in terms of diarization error rate (%) for ODESSA database when using differently trained SAD, SCD, and EMB models.

SAD model	SCD model	EMB model	Hyper-params	DER (%)
AMI	AMI	AMI	AMI	40.2
AMI	AMI	VoxCeleb	AMI	36.8
AMI	AMI	AMI	ODESSA	15.6
AMI	AMI	VoxCeleb	ODESSA	15.9

5 Low-latency speaker spotting

An automatic speaker verification (ASV) system is usually tasked with determining whether or not an audio sequence contains a given speaker [13]. Almost all work in the area involves offline processing [14, 15, 16, 17], whereas many practical applications require some form of online processing. This report presents the still ongoing work to develop a somewhat different system. In the proposed task, the ASV system is required to determine whether or not an audio sequence contains a given speaker *as quickly as possible*.

The motivation relates to the needs of the security services. These involve the rapid and efficient detection of known speakers from high volume audio streams. In such cases, rapid detection is needed in order to facilitate rapid reaction or response to potentially hostile intent; the first step subsequent to detection involves a security agent listening immediately to the audio stream.

In this application the cost of missing known speakers is high and the available resources to support human listening are limited. In this sense, the appropriate metric for the assessment of solutions is similar to that used in the majority

of related research [18], namely the cost of detection (C_{det}). Here though, the emphasis on low-latency necessitates a two-dimensional metric which combines the cost of detection with the detection lag.

The minimisation of the detection lag has implications on the manner in which an audio sequence is processed. Contrary to the majority of related research, the low-latency speaker spotting task implies processing at a segmental level. While shorter segments will allow for detection with shorter lags, the associated reduction in data will naturally degrade reliability [19]. Furthermore, in our application there is also potential for multiple, competing speakers. Here too, then, there are differences between the existing research and the low-latency speaker spotting task. Solutions will likely combine ASV technology with some form of speaker diarization.

While speaker diarization has also reached a certain level of maturity, there are differences between the usual approach and that needed for our application. While diarization is typically performed at the segmental level, most approaches cluster segments across an entire audio recording. The modest amount of work which has looked at on-line or low-latency speaker segmentation and speaker diarization is, however, well-suited to our task.

5.1 Problem definition

The low latency speaker spotting (LLSS) task is illustrated in Figure 2. It illustrates the sequence of an audio stream (*e.g.* an intercepted telephone conversation) during which a known, target speaker (for which example speech data is available) is active during the indicated segments. The target is active from time t^* but is detected only at time t_θ . The goal of the LLSS task is to detect the activity of the target speaker as soon as possible, i.e. to minimise the detection latency $t_\theta - t^*$.

An LLSS system should thus output regular log-likelihood estimates (blue profile in Figure 2) according to:

$$\Lambda(t) = \ln f(a_0^t|H_0) - \ln f(a_0^t|H_1)$$

where a_0^t is the audio from time $t = 0$ to time t and $f()$ is a conditional probability density given hypothesis H_0 or H_1 , namely that the target speaker is either active in the audio segment or not. Given a detection threshold θ , a perfect LLSS system should then return $\Lambda(t) < \theta$ for $t < t^*$ and $\Lambda(t) \geq \theta$ for $t \geq t^*$.

In practice estimates of the log-likelihood ratio need not be produced periodically, but can be produced at arbitrary instances, leading to piecewise constant functions $\Lambda : \mathbb{R}^+ \mapsto \mathbb{R}$.

5.2 Evaluation metrics

An ideal LLSS system would trigger an alarm as soon as the target speaker starts speaking. In practice, this is not feasible as a certain amount of speech from the target speaker is needed before being able to recognize them.

For instance, in Figure 2, the alarm is triggered at $t_\theta \approx 150s$ while the target speaker starts speaking at $t^* \approx 100s$, leading to an *absolute latency* δ of approximately 50s. Different values of θ lead to different latencies. Low values of θ lead to the alarm being triggered too early (in which case latency is arbitrarily set to 0). High values of θ lead to the alarm not being triggered at all. In between,

latency increases monotonically with θ . More precisely, the *absolute latency* is defined as

$$\delta_\theta = \max(t_\theta - t^*, 0) \quad (1)$$

where t^* is the time when target starts speaking and t_θ is the time when the alarm is first triggered (arbitrarily set to T^* , the last time speech has been detected, in case it is never triggered):

$$t_\theta = \begin{cases} \arg \min_{t \in \mathbb{R}^+} \Lambda(t) > \theta & \text{if } \exists t \in \mathbb{R}^+, \Lambda(t) > \theta \\ T^* & \text{otherwise} \end{cases} \quad (2)$$

However, this definition may lead to arbitrarily high latency in case a first (possibly short) utterance of the target speaker is missed and the second utterance happens long after. A more realistic alternative metric is the *speaker latency*, defined as the actual duration of speech uttered by the target speaker in the $[t^*, t_\theta]$ time range.

Depending on the final application, we might want to evaluate the detection performance of a LLSS system at a given application-driven latency δ . In this *fixed latency* scenario, the system is expected to trigger an alarm during the $[0, t^* + \delta]$ time range. Only those scores are considered to evaluate the detection performance of the system:

$$\lambda_\delta = \max_{t \in [0, t^* + \delta]} \Lambda(t) \quad (3)$$

Depending on the value of the detection threshold θ , the system will trigger an alarm if $\lambda_\delta \geq \theta$ or will not if $\lambda_\delta < \theta$. Standard speaker recognition metrics can then be reported, including false alarm rate $FAR_\delta(\theta)$, false rejection rate $FRR_\delta(\theta)$, equal error rate EER_δ , and detection cost $C_{det}^\delta(\theta)$:

$$C_{det}^\delta(\theta) = C_{miss} \times P_{target} \times FRR_\delta(\theta) + C_{false\ alarm} \times (1 - P_{target}) \times FAR_\delta(\theta) \quad (4)$$

In a *variable latency* scenario, one may also let the system use whichever latency gives the best detection performance (*i.e.* $\delta = \infty$). As depicted in the rightmost plot of Figure 4, detection performance and detection latency are then two complementary (and possibly contradictory) metrics. The average detection latency increases monotonically with θ , while the detection cost reaches its minimum value for a specific value of θ . Therefore, one may rely on such $C_{det} = f(\delta)$ curves to compare different systems.

5.3 Experimental protocol

The evaluation of LLSS solutions requires a large database of multi-speaker audio recordings and ground-truth speaker and segment level annotations. While several multi-speaker databases exist (e.g. the SITW database [20]), the Augmented Multi-party Interaction (AMI) meeting corpus [21] is the only publicly available database provided with speaker and segment annotations. Consequently, it was adopted for all experimental work reported in this project.

Despite the use of a standard database, it was necessary to design new protocols to support the development and evaluation of LLSS solutions. Nonetheless,

Table 3: LLSS protocol details: number of speakers, number of enrolled models, and number of target and non-target trials in AMI database.

Subset	# speakers	# models	# target	# non-target
Train	127	-	-	-
Dev.	22	89	771	3857
Eval.	24	106	994	5366

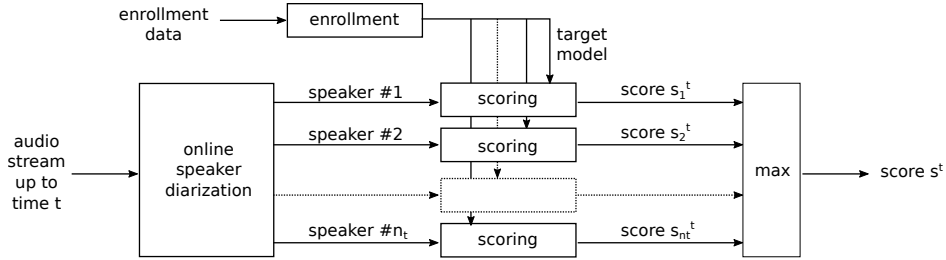


Figure 3: Common architecture of all baselines.

the standard full-corpus⁷ training, development and evaluation partition is still respected. Training data is used exclusively for background modelling. Speaker disjoint development and evaluation sets are both partitioned into enrollment and test subsets and are furthermore split into shorter 10-minute sub-sessions. Enrollment data is used to train target speaker models. The single file containing the greatest amount of speech per target speaker was split into 60-second segments, with different speaker models being learned from each. While the splitting of audio files in this way is not ideal, it serves to maximize the number and variability of trials and to provide for better generalised diarization.

A single LLSS trial is similar in nature to a classical ASV trial; it involves an enrolled target model, a test sub-session, and a trial class (target/non-target). Target trials for a given speaker are defined by using all the test sub-sessions in which the target speaker is active. Remaining sub-sessions correspond to non-target trials. The protocol described above results in the number of speakers, models target and non-target trials illustrated in Table 3.

5.4 Baselines

In this section, we describe several baseline solutions to the LLSS task, that all share a common architecture depicted in Figure 3. At any time t , online speaker diarization provides a set of n_t speaker clusters $\{c_i^t\}_{1 \leq i \leq n_t}$. The scoring backend is then used to obtain the score (or likelihood-ratio) s_i^t for all speech in each cluster c_i^t with respect to the target speaker model ω^* . The final score at time t is defined as the maximum score over all clusters: $s^t = \max_{1 \leq i \leq n_t} s_i^t$.

Speaker diarization should benefit the LLSS task by providing it with pure speaker segments and accumulating all previous speech from the current speaker before the comparison with the target. Let $C_t = \{c\}$ be the set of speaker clusters

⁷groups.inf.ed.ac.uk/ami/corpus

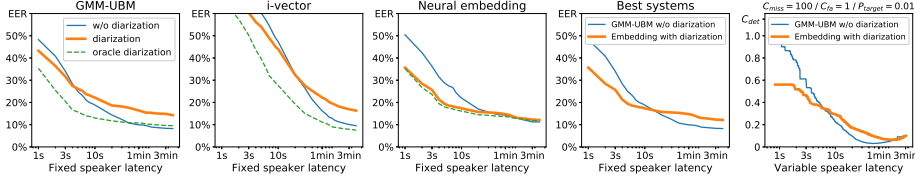


Figure 4: Influence of the detection latency on the detection performance (EER or C_{det}) for AMI database.

resulting of an online speaker diarization system at time t and $\sigma(c, \omega^*)$ the score or likelihood-ratio for all speech belonging to a cluster c relative to the target speaker model ω^* . Then $\tilde{\Lambda}(t) = \max_{c \in C_t} \sigma(c, \omega^*)$.

We compare two online speaker diarization modules. They both rely on an initial LSTM-based voice activity detector [1]. The first online diarization module does not perform any clustering: it just relies on a 3s sliding window (with a 1s shift) and creates a new cluster at each step. It is therefore denoted as “*without diarization*” in the rest of the report. The actual “*automatic diarization*” system is based on i-vectors [17] and sequential clustering of the same sliding window using the cosine similarity and an empirically optimized threshold to assign segments to existing clusters, or to create new ones. Speaker clusters are represented by i-vectors extracted from the averaged sufficient statistics of their respective segments. The system uses 19 MFCC coefficients as frontend, a universal background model (UBM) of 256 components and a T matrix of rank 100, both learned on the training data. We also report results using “*oracle diarization*” in order to estimate the impact of diarization errors on the overall LLSS systems.

We use three different backends for speaker modeling and comparison: GMM-UBM, i-vector, and neural embedding.

GMM-UBM. This backend is a standard, 256-component Gaussian mixture model with universal background model (GMM-UBM) [14], *maximum a posteriori* (MAP) speaker enrollment, and log-likelihood ratio scoring.

i-vector. This backend is an i-vector system [17] with a T matrix of dimension 100 and cosine similarity scoring between target speaker and test i-vectors. Both GMM-UBM and i-vector systems share the same MFCC frontend and UBM as the automatic diarization system.

Neural embedding. This backend is based on the neural speaker embedding approach introduced in [22] and further improved in [23]. In a nutshell, an LSTM-based neural network is trained to project 3.2s speech sequences into a 192-dimensional space, using the triplet loss paradigm. Implementation details are identical to the ones used in [24]. The target (resp. cluster) model is the sum all embeddings extracted from a sliding window with a 0.8s shift over the enrollment data (resp. cluster). Resulting vectors are compared directly using the cosine distance.

Table 4: Equal error rate (%) at fixed speaker latency for AMI database

Backend	Diar.	3s	10s	30s	1min
GMM		31.0	12.0	6.3	4.4
UBM	✓	29.6	19.2	14.3	12.2
Neural embedding	✓	23.5	10.0	7.4	6.2
		17.7	9.0	7.2	6.0

5.5 Low-latency speaker spotting results for AMI dataset

Leftmost plots in Figure 4 display the evolution of the EER on the test set as a function of the fixed speaker latency, for each combination of diarization and backend configurations. Whichever the backend, the same general trends can be observed: for very low latencies up to a few (tens of) seconds, a diarization system brings a clear benefit. This advantage is much more pronounced for the oracle configuration, showing room for improvement of the automatic diarization system: it indeed presents a diarization error rate of 44.2% using a standard collar of 250ms, with a 60.1% purity and 65.1% coverage. Finally, the differences vanish when considering latencies above one minute – but we believe that this latency level is far beyond the use case needs.

Table 4 details the EER figures for the two best performing backends and for typical latency values: at 3s and 10s latencies, the neural embedding with automatic diarization performs best at 17.7% and 9.0% respectively; the GMM-UBM system without diarization then takes the lead at 30s.

A measure of the variable speaker latency is shown in the rightmost plot of Figure 4 where C_{det} is reported with usual costs from the NIST evaluations. Obviously, selecting the system with minimal C_{det} is not a winning strategy in a LLSS task; instead one needs to carefully balance between both performance and latency constraints, e.g. selecting the lowest average latency for an admissible cost.

5.6 Low-latency speaker spotting results for ODESSA database

Since ODESSA database is smaller than AMI in size, the database is treated as one *Test* set. Therefore, in all experiments with ODESSA database, the models trained on other databases were used. Also, to train the hyper-parameters of the ‘Yin2018’ diarization system, AMI’s *Train* set was used. This approach reflects the practical scenario on evaluating the system on ‘unseen’ data.

The same low-latency speaker spotting system was applied to ODESSA database as for AMI dataset (see Section 5.5. Table 5 shows the EER for the two systems with neural embeddings trained on AMI and VoxCeleb database, since neural embeddings based system performed the best for the shorter speaker latencies in Table 4. Also, since ODESSA database contains shorter conversations, the speaker latency for which we have computed EER values ranges from 1s to 15s. From the Table 5, we can note how it is challenging for current state of the art systems to generalize on the unseen data, since the EER values are significantly higher for ODESSA compared to the values obtained on the Test set of the same database the system was trained on as in Table 4.

Table 5: EER for different latencies of segmental LLSS system computed on ODESSA database.

Embeddings	1s	3s	5s	10s	15s
AMI	78.51	72.93	48.43	33.59	31.09
VoxCeleb	78.83	72.62	50.23	34.66	31.80

6 Reproducible research

The datasets, experiments, source code, and the documentation to reproduce the experiments of this project are available on GitLab³.

The reproducibility is also facilitated by using *pyannote.audio* open source framework that relies on the use of *pyannote.metrics* [5] and configuration files, while strict enforcement of train/dev/eval split with *pyannote.database* ensures machine learning good practices.

AMI, CallHome, CallHomeSRE, REPERE, ESTER, and the in-house created databases are all publicly available and have the corresponding *pyannote* packages available for them (see the project documentation³). The cross-database evaluation³, the proposed LLSS protocol⁸, and corresponding evaluation metrics *pyannote.metrics* [5] are available as open-source software Python packages. Finally, the code for the baselines is available for public use.

7 Conclusions

This report describes ODESSA project and Idiap’s contributions to each essential parts of the project. The database of VoIP data was collected and contributions were made to the work on diarization (based on open source *pyannote.audio* framework), including the cross-database evaluations, and low-latency speaker spotting.

References

- [1] 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.
- [2] G. Gelly and J.-L. Gauvain, “Optimization of RNN-Based Speech Activity Detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, pp. 646–656, March 2018.
- [3] L. Bullock, H. Bredin, and L. P. Garcia-Perera, “Overlap-Aware Resegmentation for Speaker Diarization.” Submitted to ICASSP 2020.
- [4] R. Yin, H. Bredin, and C. Barras, “Neural Speech Turn Segmentation and Affinity Propagation for Speaker Diarization,” in *Proc. Interspeech 2018*, pp. 1393–1397, 2018.

⁸<https://gitlab.eurecom.fr/odessa/llss>

- [5] H. Bredin, “pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems,” in *Proc. Interspeech 2017*, (Stockholm, Sweden), August 2017.
- [6] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” in *Proc. InterSpeech 2018*, 2018.
- [7] J. Carletta, “Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus,” *Language Resources and Evaluation*, vol. 41, no. 2, 2007.
- [8] G. Gravier, G. Adda, N. Paulson, M. Carré, A. Giraudel, and O. Galibert, “The ETAPE Corpus for the Evaluation of Speech-based TV Content Processing in the French Language,” in *Proc. LREC 2012*, 2012.
- [9] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, “The Second DIHARD Diarization Challenge: Dataset, Task, and Baselines,” in *Proc. Interspeech 2019*, pp. 978–982, 2019.
- [10] A. Giraudel, M. Carré, V. Mapelli, J. Kahn, O. Galibert, and L. Quintard, “The REPERE corpus : a multimodal corpus for person recognition,” in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, (Istanbul, Turkey), pp. 1102–1107, European Languages Resources Association (ELRA), May 2012.
- [11] S. Galliano, E. Geoffrois, G. Gravier, J. f. Bonastre, D. Mostefa, and K. Choukri, “Corpus description of the ester evaluation campaign for the rich transcription of french broadcast news,” in *In Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC 2006)*, pp. 315–320, 2006.
- [12] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.
- [13] T. Kinnunen and H. Li, “An overview of text-independent speaker recognition: From features to supervectors,” *Speech Communication*, vol. 52, no. 1, pp. 12–40, 2010.
- [14] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.
- [15] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, “Support vector machines using gmm supervectors for speaker verification,” *IEEE Signal Processing Letters*, vol. 13, pp. 308–311, May 2006.
- [16] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, “Joint factor analysis versus eigenchannels in speaker recognition,” *IEEE Trans. on Audio, Speech, and Language Proc.*, vol. 15, pp. 1435–1447, May 2007.
- [17] N. Dehak, P. Kenny, R. Dehak, P. Ouellet, and P. Dumouchel, “Front-end factor analysis for speaker verification,” *IEEE Trans. on Audio, Speech and Language Proc.*, vol. 19, pp. 788–798, 2011.

- [18] M. A. Przybocki, A. F. Martin, and A. N. Le, “NIST speaker recognition evaluation chronicles - part 2,” in *Proc. Odyssey*, pp. 1–6, June 2006.
- [19] A. Sarkar, D. Matrouf, P.-M. Bousquet, and J.-F. Bonastre, “Study of the effect of i-vector modeling on short and mismatch utterance duration for speaker verification,” in *Proc. Interspeech*, 2012.
- [20] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, “The 2016 speakers in the wild speaker recognition evaluation,” in *Proc. Interspeech*, pp. 823–827, 2016.
- [21] J. Carletta, “Unleashing the killer corpus: experiences in creating the multi-everything ami meeting corpus,” *Language Resources and Evaluation*, vol. 41, no. 2, pp. 181–190, 2007.
- [22] H. Bredin, “TristouNet: Triplet Loss for Speaker Turn Embedding,” in *Proc. IEEE ICASSP*, March 2017.
- [23] G. Gelly and J.-L. Gauvain, “Spoken Language Identification using LSTM-based Angular Proximity,” in *Proc. Interspeech*, August 2017.
- [24] G. Wisniewski, H. Bredin, G. Gelly, and C. Barras, “Combining Speaker Turn Embedding and Incremental Structure Prediction for Low-Latency Speaker Diarization,” in *Proc. Interspeech*, August 2017.