Low-latency speaker spotting with online diarization and detection

Jose Patino[1], Ruiqing Yin[2], Héctor Delgado[1], Hervé Bredin[2], Alain Komaty[3], Guillaume Wisniewski[2], Claude Barras[2], Nicholas Evans[1] and Sébastien Marcel[3]

[1] EURECOM, France
[2] LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, France
[3] Idiap, Switzerland

Abstract

This paper introduces a new task termed low-latency speaker spotting (LLSS). Related to security and intelligence applications, the task involves the detection, as soon as possible, of known speakers within multi-speaker audio streams. The paper describes differences to the established fields of speaker diarization and automatic speaker verification and proposes a new protocol and metrics to support exploration of LLSS. These can be used together with an existing, publicly available database to assess the performance of LLSS solutions also proposed in the paper. They combine online diarization and speaker detection systems. Diarization systems include a naive, over-segmentation approach and fully-fledged online diarization using segmental i-vectors. Speaker detection is performed using Gaussian mixture models, i-vectors or neural speaker embeddings. Metrics reflect different approaches to characterise latency in addition to detection performance. The relative performance of each solution is dependent on latency. When higher latency is admissible, i-vector solutions perform well; embeddings excel when latency must be kept to a minimum. With a need to improve the reliability of online diarization and detection, the proposed LLSS framework provides a vehicle to fuel future research in both areas. In this respect, we embrace a reproducible research policy; results can be readily reproduced using publicly available resources and open source codes.

1. Introduction

An automatic speaker verification (ASV) system is usually tasked with determining whether or not an audio sequence contains a given speaker [1, 2]. Almost all work in the area, e.g. [3, 4, 5, 6], involves offline processing. This paper reports our ongoing work to develop a somewhat different system. In our task the ASV system is required to determine whether or not an audio sequence contains a given speaker *as quickly as possible*. We refer to this task as low-latency speaker spotting (LLSS).

The motivation relates to the needs of the security and intelligence services. These involve the rapid and efficient detection of known, target 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 an agent listening immediately to the audio stream. While it is not the focus of our work, the LLSS task also relates to civilian and consumer applications involving voice-based personal assistants and speaker-dependent, but text-independent wake-up systems.

For the security/intelligence application, the cost of missing target 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 [7, 8], namely the cost of detection (C_{det}) with the usual parameters. Here though, the emphasis on low-latency necessitates a two-dimensional metric which combines the cost of detection with the detection lag or latency.

The minimisation of latency has implications on the manner in which an audio sequence is processed. The LLSS task implies processing at a segmental level. While shorter segments will allow for detection with shorter latency, the associated reduction in data will naturally degrade reliability [9], inferring the need to strike a balance between latency and reliability. Furthermore, in our application there is also potential for multiple, competing speakers. Here too, then, there are differences between the existing research and the LLSS task. Solutions will likely combine ASV technology with some form of online segmentation or speaker diarization.

The contributions of this paper include: (i) a formulation of the LLSS task and metrics in Section 3; (ii) the first LLSS solutions in Section 4; (iii) a protocol for LLSS assessment in Section 5; (iv) the benchmarking of LLSS solutions against oracle solutions in Section 6. The database used for this work is based upon the adaptation of a database that is already in the public domain. In embracing a policy of reproducible research, the metrics and protocol used for the work reported in this paper are being released publicly, together with open-source versions of the presented LLSS solutions.

2. Prior work

The topics of speaker diarization and automatic speaker verification are closely related to the LLSS task. Speaker diarization [10] involves the clustering of speech recordings into speaker-homogeneous segments. In contrast to the LLSS task, speaker diarization is typically performed offline and with no prior information (*e.g.* number of speakers or speaker models). A number of online diarization [11, 12, 13, 14, 15] and speaker tracking [16, 17] solutions have been reported. These use online speaker clustering algorithms [18, 19]. Only speaker tracking systems assume prior knowledge of target speakers but they do not consider latency.

Ideally, speaker recognition should be possible by using small amounts of speech. Unfortunately, with current technology, this is only possible if the text employed for enrolment and

This work was supported through funding from both the Agence Nationale de la Recherche (French research funding agency) and the Swiss National Science Foundation within the ODESSA (ANR-15-CE39-0010) and PLUMCOT (ANR-16-CE92-0025) projects.

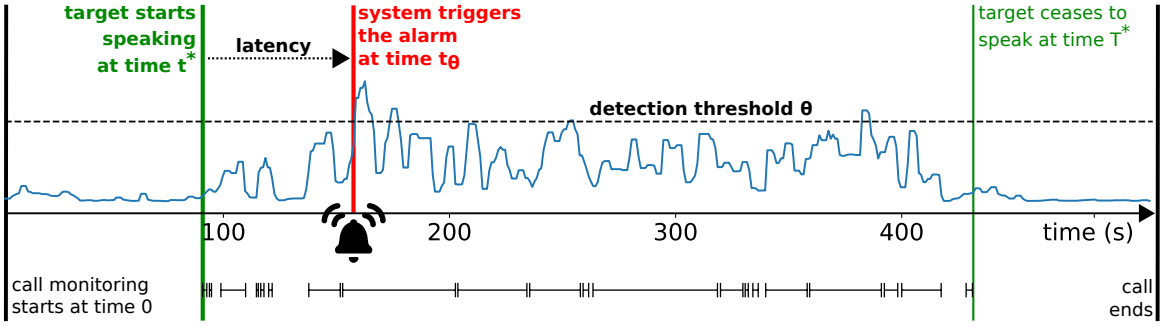

Figure 1: Low-latency speaker spotting (LLSS) systems aim to detect target speakers with the lowest possible latency.

testing phases is constrained. This task is known in the literature as text-dependent speaker recognition [20, 21], and is often associated with specific applications, *e.g.* user-friendly human-robot interaction [22]. On-going research focused on keyword-spotting offers solutions that do not require more than a mere few seconds of speaker content [23], resulting in extremely low latencies.

However, text-related constraints are not suitable for certain scenarios, like surveillance, which motivate the majority of text-independent ASV research [1]. The text-independent ASV task tends to involve either single-speaker or two-speaker recordings. Research within the scope of the Speakers in the Wild (SITW) [24] initiative considers multi-speaker scenarios which necessitate some form of diarization as a precursor to ASV. Published research addresses only offline processing and the lack of speaker segmentation references means that the SITW database is ill-suited to the exploration of LLSS.

None of the prior work addresses all aspects of the LLSS task. Existing databases do not support the joint evaluation and optimisation of speaker diarization and text-independent recognition, nor the development of online, low-latency solutions. In addition, while existing databases can be adapted, there are no common protocols to support LLSS research.

3. Low-latency speaker spotting

This section provides a formal definition of the low-latency speaker spotting (LLSS) task and outlines two different approaches to evaluate the latency achieved by potential solutions.

3.1. Task definition

The low latency speaker spotting (LLSS) task aims at determining whether or not an audio sequence contains a given speaker with the shortest possible delay. Figure 1 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 $t_\theta - t^*$.

Note that this is different from explicitly providing the speaker starting time t^*. If this value is needed in the context of a specific application, further automatic or manual processing may occur in order to refine t^* estimates once the target speaker has been detected. For a monitoring system, the audio stream may have been buffered and a security agent may listen to the stream after rewinding the audio by a few seconds, according to the typical detection latency; for a real-time human-robot interaction application, spotting the various users as quickly as possible is the main goal, regardless of the precise value of t^*.

An LLSS system may typically rely on regular log-likelihood ratio estimates (example blue profile in Figure 1) according to:

$$\Lambda(t) = \ln f(a_0^t|H_1) - \ln f(a_0^t|H_0) \quad (1)$$

where a_0^t is the audio from time $t = 0$ to time t and $f()$ is a conditional probability density given hypothesis H_1 or H_0, namely that the target speaker is either active in the audio stream at some point up to time t or not.

Given a detection threshold θ, the LLSS decision Γ at time t would then be:

$$\Gamma(t) = \mathbb{1}\left(\max_{\tau\in[0,t]} \Lambda(\tau) - \theta\right) \quad (2)$$

where $\mathbb{1}$ is the Heaviside function (returning 0 and 1 for negative and positive values, respectively). Note that the decision is irreversible once the threshold has been reached, even if Λ may later decrease. An ideal log-likelihood ratio estimator should thus return $\Lambda(t) < \theta$ for $t < t^*$ and $\Lambda(t) \geq \theta$ for $t \geq t^*$. In practice, $\Lambda(t)$ need not be produced periodically, but can be produced at arbitrary instances, leading to piecewise constant functions $\Lambda : \mathbb{R}^+ \mapsto \mathbb{R}$.

3.2. Absolute vs. speaker latency

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 they can be recognised or 'spotted'. For instance, in Figure 1, 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 $50s$.

In practice, the absolute latency δ will be influenced by the detection threshold θ. Low values of θ may lead to the alarm being triggered too early, before the target speaker starts speaking. For the sakes of evaluation (specifically the need to maintain a constant number of trials and to assign a latency to each), those trials are not marked as false alarms. Instead, their latency is bound to 0[1]. High values of θ may lead to the alarm not being triggered at all. In between, latency will likely increase monotonically with θ.

[1]Note that low values of θ would also lead to a high number of false alarms, making the system useless in practice. Such operating points lack practical interest.

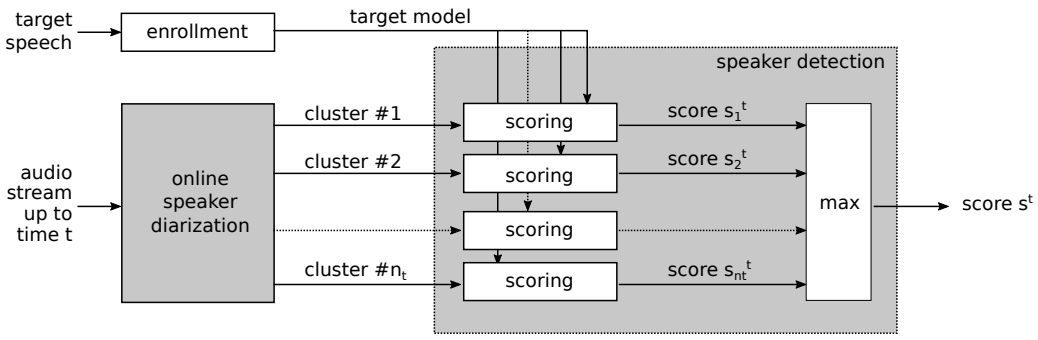

Figure 2: Common architecture to proposed LLSS solutions

More precisely, the *absolute latency* is defined as:

$$\delta_\theta = \max(t_\theta - t^*, 0) \tag{3}$$

where t^* is again the time at which the target starts speaking for the first time and t_θ is the time at which the alarm is first triggered.

In the case that the alarm is never triggered, t_θ is set to the time T^* in the audio stream at which the target speaker ceases to be active, giving:

$$t_\theta = \begin{cases} \min\left\{t \in \mathbb{R}^+ | \Lambda(t) > \theta\right\} & \text{if } \exists t \in \mathbb{R}^+,\ \Lambda(t) > \theta \\ T^* & \text{otherwise} \end{cases} \tag{4}$$

However, this definition may lead to arbitrarily high latency in the case, for example, that the first (possibly short) utterance of the target speaker is missed and the second utterance occurs long after. A more meaningful, 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.

3.3. Detection under variable or fixed latency

For a given detection threshold θ, the value of either the absolute or the speaker latency δ_θ as defined in Eq. (3) will depend on the actual trial. If one does not constrain the maximal latency and lets the system use whichever latency gives the best detection performance (*i.e.* equivalent to $\Gamma(t)$ with $t \to \infty$), then this is referred to as a *variable latency* scenario. Detection performance and detection latency are then two complementary (but 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 curves displaying the detection cost as a function of δ to compare the performance of different systems.

However, averaging the latency across trials may in fact hide very different behaviours. Depending on the final application, we might prefer 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. The detection performance of such a system may then be calculated using corresponding scores according to:

$$\lambda_\delta = \max_{t \in [0, t^* + \delta]} \Lambda(t) \tag{5}$$

Depending on the value of the detection threshold θ, the system will trigger an alarm if $\lambda_\delta \geq \theta$ whereas no alarm will be triggered if $\lambda_\delta < \theta$. Standard speaker recognition metrics then apply. They include the false alarm rate $FAR_\delta(\theta)$, the missed detection rate $MDR_\delta(\theta)$, the equal error rate EER_δ, and the detection cost $C^\delta_{det}(\theta)$ given by:

$$\begin{aligned} C^\delta_{det}(\theta) \quad = \quad & C_{\text{miss}} \times P_{\text{target}} \times MDR_\delta(\theta) + \\ & C_{\text{false alarm}} \times (1 - P_{\text{target}}) \times FAR_\delta(\theta) \end{aligned} \tag{6}$$

4. LLSS solutions

This section describes a number of different solutions to the LLSS task. They share a common architecture depicted in Figure 2 which combines online speaker diarization with different approaches to speaker detection. At any time t, online speaker diarization provides a set of n_t speaker clusters $\{c_i^t\}_{1 \leq i \leq n_t}$. Speaker detection is then applied to compare the speech segments in each cluster c_i^t against a set of pre-trained target speaker models, thereby giving scores (or likelihood-ratios) s_i^t. A final score at time t is defined as the maximum score over all clusters: $s^t = \max_{1 \leq i \leq n_t} s_i^t$. The remainder of this section describes the two different online speaker diarization systems and three speaker detection systems explored in this work.

4.1. Online speaker diarization

Two different approaches to online speaker diarization are compared. Both rely on an LSTM-based voice activity detector (VAD) [25].

Segmental diarization: the first online diarization module does not perform any clustering: it relies simply on a segmental approach of a 3s sliding window with a 1s shift, and creates a new cluster at each step. Note that only speech content, often shorter than the complete 3s, is considered. This approach is denoted as segmental diarization in the rest of the paper.

Automatic diarization: the second automatic system is based on i-vectors [6] and online sequential clustering using the same sliding window, a cosine similarity measure and an empirically optimized threshold to assign segments to existing clusters, or to create new ones. Should the score of a new segment produced from its comparison against the set of existing clusters fall below the threshold, then it will be assigned to a new cluster. Otherwise, it will be assigned to the cluster among the existing set corresponding to the highest score. Speaker clusters are represented by i-vectors extracted from the averaged sufficient statistics of their respective segments. The system uses 19 MFCC coefficients as a frontend, a universal background model (UBM) of 256 components and a T matrix of rank 100, both

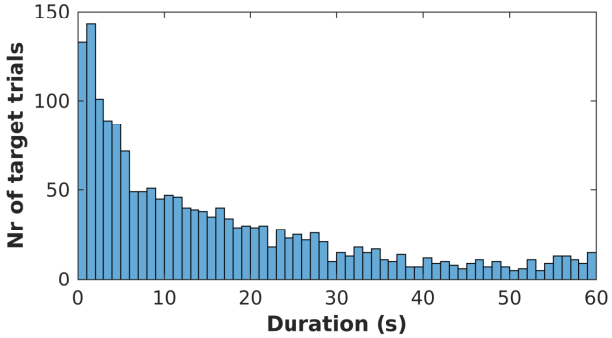

Figure 3: Distribution of target speech duration per trial for the designed test subset.

learned from training data. i-vectors are length-normalised and whitened. All parameters were empirically optimised on the development set with according to the standard diarization error rate (DER) metric.

Oracle diarization: the performance of both online diarization systems is compared to that of an oracle diarization system in order to observe the impact of diarization errors on LLSS performance. The oracle system simulates the behaviour of an error-less, but still *online* system; it uses data from time zero to time t.

4.2. Speaker detection

The performance of three different approaches to speaker detection were explored. The systems considered are described in the following.

GMM-UBM: the first system is a standard, 256-component Gaussian mixture model with universal background model (GMM-UBM) [3], with a conventional MFCC frontend (the same as that used for diarization), *maximum a posteriori* model adaptation and log-likelihood ratio scoring.

i-vector: the second is an i-vector system [6] with a T matrix of dimension 100 and PLDA scoring [26] between target and test i-vectors, that uses a 100-dimensional speaker space and was trained on the same data as the UBM and the total-variability matrix. The frontend features are the same as that of the GMM-UBM system and the diarization system.

Neural embedding: the final system is based on the neural speaker embedding approach introduced in [27] and further improved in [28]. Briefly, an LSTM-based neural network is trained to project speech sequences into a 192-dimensional space, using the triplet loss paradigm. Implementation details are identical to the ones used in [29]. The target (resp. cluster) model is the sum all embeddings extracted from a 3s sliding window with a 1s shift over the enrollment data (resp. cluster). Resulting vectors are compared using the cosine distance.

5. LLSS assessment

None of the existing databases employed in either speaker diarization or speaker detection/verification are suited to the exploitation of the LLSS task. This section describes the steps taken to adapt an existing database for this purpose.

Table 1: LLSS protocol details: number of speakers, number of enrolled models, and number of target and non-target trials.

Set	# speakers	# models	# target	# non-target
Train	127	-	-	-
Dev.	22	121	9430	64451
Eval.	24	164	12250	102560

5.1. Database

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 [24]), the Augmented Multi-party Interaction (AMI) meeting corpus [30] is widely used, publicly available and is provided with the necessary speaker and segment annotations. Consequently, it was adopted for all experimental work reported in this paper.

The AMI database contains a set of audio meetings containing sessions of approximately 40 minutes and recorded across 3 different sites under different conditions and scenarios. As a consequence, speakers groups are disjoint in terms of site, while meetings collected at each site contain independent speaker groups with around 4 speakers each. There are approximately 4 meeting recordings for each group.

5.2. Protocols

Despite the use of a standard database, it was necessary to design new protocols to support the development and evaluation of LLSS solutions. Nonetheless, the standard full-corpus[2] training, development and evaluation partition is still respected. All experiments were performed using data corresponding to the mix-headset condition of the AMI meeting corpus.

Training data is used exclusively for background modelling. Speaker disjoint development and evaluation sets are both partitioned into enrollment and test subsets. Enrolment data is used to train target speaker models.

The single session which contains the greatest amount of speech from a given target speaker is used for enrolment. The speech from the target speaker is divided into N 60-second, overlap-free speech segment splits. A subset of these N splits is randomly selected as the data for the M different models generated for the target speaker. Since N varies across target speakers (due to varying quantities of data per speaker), M is set to the median of every N for each target speaker.

Testing content is generated from all the non-enrolment content for each given speaker and through sub-session splits of 1-minute duration. Each split contains speech from 0 to 4 speakers. While not ideal, under strict data constraints, the splitting of audio files serves to increase the number of trials and variability.

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. This leads to a distribution of target trials illustrated in Figure 3. The target speaker content per trial exceeds only rarely 30 seconds duration. Remaining sub-sessions

[2] `groups.inf.ed.ac.uk/ami/corpus`

correspond to non-target trials. The protocol described above results in the number of speakers, models, target and non-target trials illustrated in Table 1.

6. Experimental results

The performance of the proposed LLSS solutions is analysed in two different manners. The first analysis is in terms of fixed and variable speaker latency using detection metrics described in Section 3.3. Second, we analyse diarization influences upon LLSS performance.

6.1. LLSS performance: fixed latency

Plots in Figure 4 depict the evolution in EER for the evaluation set as a function of *fixed speaker latency* (latter part of Section 3.3). Separate plots are shown for GMM-UBM, i-vector and neural embedding speaker detection solutions following either oracle, automatic or segmental diarization systems. The right-most plot compares the EER against fixed speaker latency for the best combination of diarization and detection approaches.

No matter what the detection system the best performance is observed with oracle diarization. The performance observed for automatic and segmental diarization systems is dependent upon the detection system. For the GMM system, automatic diarization fares poorly whereas for the neural embedding solution, results for automatic diarization are broadly similar to those obtained with oracle diarization.

While segmental diarization gives reasonable performance in the case of the GMM and i-vector detection systems, performance is poor for the neural embedding detection system. Discrepancies between performance for oracle, automatic and segmental diarization systems are, however, dependent to some degree on the fixed speaker latency, especially for the neural embedding detection system. While differences in performance for oracle and segmental diarization are pronounced for lower fixed latencies, these diminish almost entirely for higher fixed latencies.

Of particular interest is how latency impacts upon the performance of each LLSS solutions and then, which system performs best. A summary of the three left-most plots for the two practical approaches to diarization (segmental or automatic only - the performance of the oracle system is discounted) is illustrated in the right-most plot of Fig. 4. It shows that the neural embedding detector outperforms the GMM and i-vector systems by a significant margin for the lowest fixed latency of 3s. For higher fixed latencies, however, the GMM and i-vector systems outperform neural embeddings, albeit by a smaller margin; there is little to choose between them.

6.2. LLSS performance: variable latency

An illustration of system performance in terms of C_{det} is depicted in Fig. 5 for *variable speaker latency* (earlier part of Section 3.3). Plots are again illustrated for each detection system and for the best corresponding diarization system (segmental or automatic). C_{det} values are determined according to the usual costs adopted by the NIST speaker recognition evaluations (SRE) [31]. Profiles in Fig. 5 show a slightly different picture than that for fixed speaker latencies, with the automatic online diarization system and neural embeddings detector showing almost universally better performance. While segmental and automatic diarization systems with GMM and i-vector detection systems show lower C_{det} between approximately 3s and 10s, differences are marginal.

In an alternative interpretation, for a given detection cost, the neural embeddings system provides shorter speaker detection latencies than GMM and i-vector systems. Obviously, selecting the system with minimal C_{det} is not necessarily a sensible strategy for a LLSS task; instead one needs to strike a balance between performance and latency constraints, *e.g.* selecting the lowest average latency for an admissible cost. In almost all cases, however, this choice remains that of the neural embeddings solution.

6.3. Diarization influences

Discrepancies in performance between segmental and automatic diarization hypotheses were initially rather puzzling. For the GMM detection system, automatic diarization performs poorly. In contrast, for the neural embedding system, automatic diarization leads to performance that is on a par with oracle diarization.

The automatic diarization system uses a form of greedy sequential clustering. When performed in an online fashion, all such systems have potential to introduce errors into the diarization hypothesis, errors from which the system can never recover. Impure clusters that contain data from more than one speaker are likely to remain impure as online diarization proceeds. Online diarization performance is illustrated in Table 2 for the evaluation partition of the AMI database. Note that DERs are naturally higher than those typically reported in the literature - those reported here relate to an *online* task. Even so, the purity of clusters it produces is reasonable, with over 70% of clusters corresponding to data from the dominant speaker. Coverage, which refers to the percentage of encountered speaker data that is assigned to the corresponding speaker model, exceeds an encouraging 80%.

It is evident that the proposed LLSS systems have different capacities to accommodate errors in the diarization hypothesis. This is mostly due to the different data demands and normalisation strategies employed by each detection solution. Referring to Figure 4, the neural embedding system copes well with data impurities. The GMM and i-vector detection systems cope less well with the same data impurities (the gap between performance for oracle and automatic diarization is greater), however the i-vector system outperforms the neural embedding system for higher latencies (albeit only marginally).

In contrast to the automatic diarization system, the segmental approach does not accumulate speaker data through clustering. Diarization performance for the segmental approach is also shown in Table 2. While results show a very high diarization error rate of 95%, they show that, as expected, purity is higher, while coverage is naturally very low. Thus, while speaker models will be comparatively poorly trained using only short segments of speech, they may yet give better performance in the case that they are trained, more often than not, using data from a single speaker. This is to be expected for such short segments since the chances of them bridging speaker turns is low. As a result, it is not necessarily surprising that the segmental diarization system performs well under some conditions. Eventually, and by pure chance, the detection system will be presented with a pure target speaker segment that will produce a high detection score.

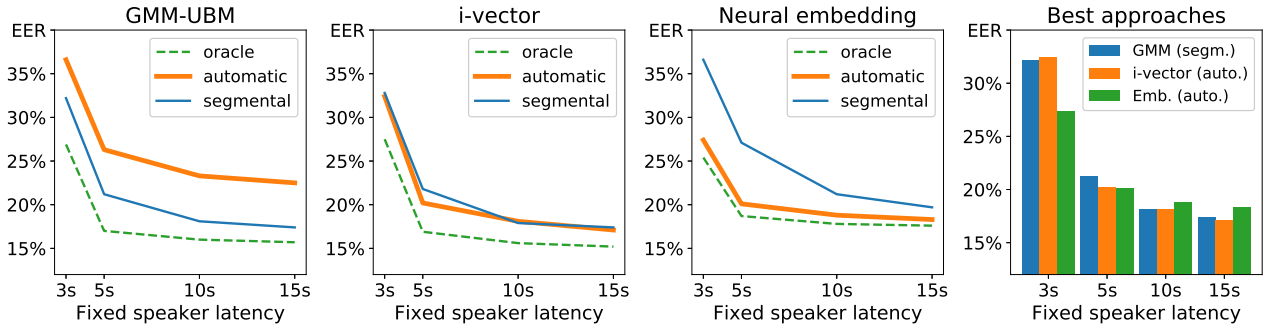

Figure 4: Influence of the detection latency on the detection performance on the evaluation set.

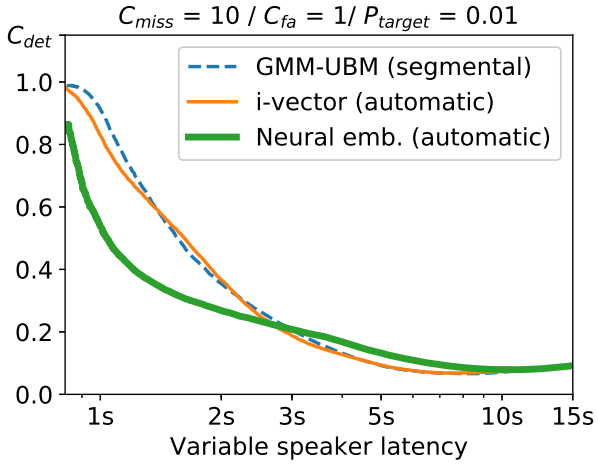

Figure 5: Detection performance as a function of the average speaker latency for the best performing automatic systems on the evaluation set.

It is clear from the analyses presented above that the dependence of detection systems upon diarization is more complex than may first appear. Future work should study this dependence further and examine more carefully the robustness to each detection solution to speaker cluster impurities. This may help to better tune the combination of online diarization and speaker detection, thus improving the reliability of LLSS solutions. The same finding may suggest that the optimisation of diarization systems with respect to the diarization error rate may not be sensible when diarization is only an enabling technology, instead of the final application.

7. Reproducible research

The AMI corpus is publicly available under a Creative Commons license[3]. The proposed LLSS protocol and corresponding evaluation metrics are available as open-source software in *pyannote.db.odessa.ami*[4] and *pyannote.metrics* [32] Python packages. Finally, baseline online diarization outputs and code for the segmental neural embedding approach are publicly available at `https://gitlab.eurecom.fr/odessa/llss`.

[3] `groups.inf.ed.ac.uk/ami/corpus`

[4] `github.com/pyannote/pyannote-db-odessa-ami`

Table 2: **Online** diarization performance in the form of DER (%), cluster purity (%) and coverage (%), obtained with the i-vector automatic online diarization system on evaluation sets, evaluated using a using a standard collar of 250ms.

Diarization system	DER	Purity	Coverage
Segmental	95.83	88.69	5.73
Automatic	34.24	75.48	81.52

8. Conclusions

This paper describes a new task termed low-latency speaker spotting (LLSS). The LLSS task is motivated by security and intelligence applications, but has application elsewhere, *e.g.* voice-based personal assistants and speaker-dependent, but text-independent wake-up systems. The LLSS task calls for the recognition of known, target speakers as quickly as possible after they become active in an audio stream.

Results show that reliable online diarization is key to minimising latency and LLSS performance overall. Differences in results obtained with oracle segmentation and segmental diarization demonstrate the challenge of automatic, online diarization; it can be difficult to outperform a simple segmental approach. Results also show differences in how speaker detection approaches cope with speaker model impurities. Together, these findings show that effective solutions to the LLSS task require a careful combination and joint optimisation of online speaker diarization and speaker detection algorithms. They also question the sense of optimising speaker diarization, online or otherwise, in isolation when diarization is only an enabling technology, instead of the end application.

Future work should investigate the differences in the behaviour of the proposed speaker detection techniques in detail. It may also investigate strategies to cope with overlapping speech from competing speakers and study more closely combined, joint optimisation of the feature extraction, online diarization and automatic speaker verification components. Emerging end-to-end approaches thus offer another avenue for future work.

9. References

[1] 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.

[2] J. H. L. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal processing magazine*, vol. 32, no. 6, pp. 74–99, 2015.

[3] 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.

[4] 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, no. 5, pp. 308–311, May 2006.

[5] 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, no. 4, pp. 1435–1447, May 2007.

[6] 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.

[7] M. A. Przybocki, A. F. Martin, and A. N. Le, "NIST speaker recognition evaluation chronicles - part 2," in *Proc. Odyssey*, June 2006, pp. 1–6.

[8] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. S. Greenberg, E. S. Reynolds, L. Mason, and J. Hernandez-Cordero, "The 2016 NIST Speaker Recognition Evaluation," *Proc. Interspeech*, pp. 1353–1357, 2017.

[9] 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.

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

[11] L. Lu and H.-J. Zhang, "Unsupervised speaker segmentation and tracking in real-time audio content analysis," *Multimedia Systems*, vol. 10, no. 4, pp. 332–343, Apr. 2005.

[12] T. Oku, S. Sato, A. Kobayashi, S. Homma, and T. Imai, "Low-latency speaker diarization based on Bayesian information criterion with multiple phoneme classes," in *Proc. IEEE ICASSP*, Mar. 2012, pp. 4189–4192.

[13] W. Zhu and J. Pelecanos, "Online speaker diarization using adapted i-vector transforms," in *Proc. IEEE ICASSP*, March 2016, pp. 5045–5049.

[14] D. Dimitriadis and P. Fousek, "Developing on-line speaker diarization system," in *Proc. Interspeech*, 2017, pp. 2739–2743.

[15] G. Soldi, M. Todisco, H. Delgado, C. Beaugeant, and N. Evans, "Semi-supervised On-line Speaker Diarization for Meeting Data with Incremental Maximum A-posteriori Adaptation," in *Proc. Odyssey*, 2016, pp. 377–384.

[16] M. Zamalloa, L.-J. Rodriguez-Fuentes, G. Bordel, M. Penagarikano, and J.-P. Uribe, "Low-latency online speaker tracking on the AMI corpus of meeting conversations," in *Proc. IEEE ICASSP*, 2010, pp. 4962–4965.

[17] M. H. Moattar and M. M. Homayounpour, "Variational conditional random fields for online speaker detection and tracking," *Speech Communication*, vol. 54, no. 6, pp. 763–780, July 2012.

[18] D. Liu and F. Kubala, "Online speaker clustering," in *Proc. IEEE ICASSP*, Apr. 2003, vol. 1, pp. 572–575.

[19] T. Koshinaka, K. Nagatomo, and K. Shinoda, "Online speaker clustering using incremental learning of an ergodic hidden Markov model," in *Proc. IEEE ICASSP*, Apr. 2009, pp. 4093–4096.

[20] A. Larcher, K. A. Lee, B. Ma, and H. Li, "Text-dependent speaker verification: Classifiers, databases and RSR2015," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[21] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE ICASSP*, 2014, pp. 4052–4056.

[22] K. A. Lee, A. Larcher, H. Thai, B. Ma, and H. Li, "Joint application of speech and speaker recognition for automation and security in smart home," in *Proc. Interspeech*, 2011.

[23] G. Heigold, I. Lopez-Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. IEEE ICASSP*, 2016, pp. 5115–5119.

[24] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The 2016 speakers in the wild speaker recognition evaluation," in *Proc. Interspeech*, 2016, pp. 823–827.

[25] R. Yin, H. Bredin, and C. Barras, "Speaker Change Detection in Broadcast TV using Bidirectional Long Short-Term Memory Networks," in *Proc. Interspeech*, August 2017.

[26] G. Sell and D. Garcia-Romero, "Speaker diarization with PLDA i-vector scoring and unsupervised calibration," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 413–417.

[27] H. Bredin, "TristouNet: Triplet Loss for Speaker Turn Embedding," in *Proc. IEEE ICASSP*, March 2017.

[28] G. Gelly and J.-L. Gauvain, "Spoken Language Identification using LSTM-based Angular Proximity," in *Proc. Interspeech*, August 2017.

[29] 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.

[30] 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.

[31] C. S. Greenberg, A. F. Martin, B. N. Barr, and G. R Doddington, "Report on performance results in the NIST 2010 speaker recognition evaluation," in *Proc. Interspeech*, 2011.

[32] H. Bredin, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Proc. Interspeech*, August 2017.