Unsupervised Speaker Diarization that is Agnostic to Language, Overlap-Aware, and Tuning Free

M. Iftekhar Tanveer[1], Diego Casabuena[1], Jussi Karlgren[2], Rosie Jones[1]

[1]Spotify, USA & [2]Spotify, Sweden

{iftekhart, diegoc, jkarlgren, rjones}@spotify.com

Abstract

Podcasts are conversational in nature and speaker changes are frequent—requiring speaker diarization for content understanding. We propose an unsupervised technique for speaker diarization without relying on language-specific components. The algorithm is overlap-aware and does not require information about the number of speakers. Our approach shows 79% improvement on purity scores (34% on F-score) against the Google Cloud Platform solution on podcast data.

Index Terms: Speaker Diarization, Sparse Optimization

1. Introduction

Speaker Diarization is the process of logging the timestamps (or writing in a diary—thus the term "diarization") when different speakers spoke in an audio recording. At Spotify, we have found that reliable speaker diarization is a foundational technology we need in order to build downstream natural language understanding pipelines for podcasts. Podcasts are more diverse in content, format, and production styles [1, 2, 3, 4] than classical application areas for diarization—requiring a revisit to the techniques.

Many established diarization techniques utilize a pipeline similar to the one shown in Figure 1(A) [5]. A typical pipeline involves several independent components. The *Embedding* component segments the audio into small chunks and converts each chunk into a vector (e.g. i-vectors [6], d-vectors [7], or x-vectors [8]) which represents the auditory characteristics of the speech heard in a chunk and the speakers who produce it. The *Clustering* component divides those vectors into several groups by applying Spectral Clustering [9, 10, 11], Agglomerative Clustering [12], or other techniques [13, 14]. Finally, a *Post Processing* component takes care of some issues not handled by the previous components, such as detecting overlapping speech with more than one simultaneous speaker [15].

Classical research in speaker diarization was focused on independently improving the performances of these blocks [5]. However, with growing popularity in the end-to-end [16] learning models using e.g. deep learning, recent implementations have used combinations of multiple modalities (e.g. audio and text) [17]. For instance, Google has developed an algorithm that performs speech recognition jointly with diarization [18]; d-vectors have been proposed to utilize both audio and text [7].

A drawback of incorporating textual information into a diarization algorithm is it also renders the algorithm dependent on language. Such an algorithm needs to be retrained for every supported language which quickly becomes impracticable[1]. Human perception is quite capable to perform speaker diarization without understanding the language, and diarization would
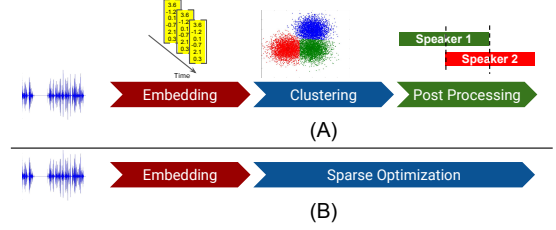

Figure 1: *(A) Classical speaker diarization approach vs. (B) our proposed approach*

seem to be a task which should be solvable automatically without specific training or specific knowledge resources for the language being spoken. In this paper, we propose an audio-only solution for speaker diarization which is naturally *language agnostic*.

The clustering component introduces several problems for the overall diarization effort. Typical clustering algorithms depend on a pre-determined number of clusters—in this application scenario, number of distinct speakers. However, it is impractical to decide the number of speakers beforehand in the speaker diarization scenario. Additionally, many clustering algorithms group the audio segments into disjoint clusters. This approach makes identifying overlapping speech (a frequent occurrence in natural and unscripted conversation, such as in podcasts) an afterthought, typically solved by introducing post-processing modules based on e.g. Hidden Markov Models [19], neural networks [20], or other methods [21]. Fully supervised speaker diarization approaches have been proposed [22], but supervision requires manual annotation of large training data sets—which is costly, time-consuming, and dreary work for human annotators. In addition, a training dataset must be balanced over various languages, production styles, and format categories (e.g. audiobooks, enacted drama, panel discussions etc.) which adds up to being a considerable engineering effort.

We propose an unsupervised speaker diarization algorithm that avoids all the problems described above by replacing the clustering and the post-processing components with a sparse (ℓ_1 regularized) optimization approach as shown in Figure 1(B). We use a pretrained audio embedding model with convenient characteristics that makes the algorithm *overlap aware*. The hyperparameters required for our proposed processing model and algorithm are automatically adjusted by employing theories from compressed sensing [23, 24] and linear algebra. Our algorithm is similar to Vipperla et al. [25] but unlike that, the proposed algorithm doesn't need any manual intervention for constructing a *global* basis for overlap handling. The embedding basis is constructed automatically by utilizing an ℓ_1-regularization over a *local* basis matrix. This approach makes it completely *tuning-free* from the users' perspective—enabling the usage in

---

large scale setting as in Spotify. The proposed solution is also highly scalable with hardware such as Graphical Processing Units (GPUs).

Finally, we compare the performance of the proposed algorithm against several industry standard solutions such as Google cloud platform and PyAnnote.

2. Methods

Our proposed approach has two major steps: a) Construction of the embedding signal, and b) Sparse optimization.

2.1. Constructing the Embedding Signal

We generate a sequence of M-dimensional ($M = 512$) vectors for an audio recording using a pre-trained VggVox [26] embedder. VggVox is a convenient choice for our purposes, for several reasons. It is trained on the VoxCeleb [26] dataset which contains over 1 million utterances for 6,112 celebrities from around the world in various contexts and situations. Due to the "wild" nature of the VoxCeleb dataset, VggVox is able to capture not only the acoustic characteristics, but also dialectal and stylistic speech characteristics. This allows us to use vector similarity metrics (such as cosine similarity) to identify whether two speech segments are from the same speaker.

VggVox vectors have the attractive and convenient characterstic of adhering to a *linearity constraint*—i.e. a VggVox embedding, $\mathbb{V}()$, of an audio chunk from speaker S_1 of length p concatenated with an audio chunk from speaker S_2 of length q, $\mathbb{V}(S_1[1:p] \oplus S_2[1:q])$ is approximately equal to its weighted arithmetic average:

$$\frac{1}{p+q}(p\mathbb{V}(S_1[1:p]) + q\mathbb{V}(S_2[1:q])) \quad (1)$$

We segment a recording into a series of overlapping chunks by sliding a 6-second window over the recording with a variable step size of 1 second or less. We set the step size to be such that it yields at least 3,600 chunks and compute a vector for each using the pretrained VggVox model.

We use MobileNet[2][27] to detect non-speech regions in the recording and we set the vectors for non-speech regions to zero. We arrange the sequence of these vectors for a recording into a matrix which we call the *embedding signal*, $\mathcal{E} \in \mathbb{R}^{M \times T}$ so that vectors from each of the T time-steps are represented in the columns of the matrix. We normalize the embedding signal, $\mathcal{E}$, such that the columns are unit vectors.

2.2. Speaker Diarization using Sparse Optimization

The embedding signal, $\mathcal{E}$, has a useful property that allows us to use techniques from compressed sensing [23, 24] for speaker diarization. The embedding signal is, in general, low rank with many dependent columns. This is due to the fact that the step size of the sequence is short and speakers do not typically change turns at a rate higher than the step size which means that the columns of $\mathcal{E}$ remain identical over several consecutive time-steps. It is possible to factor out $\mathcal{E}$ as a matrix product of an *embedding basis matrix*, $\boldsymbol{\Psi}$ and an *activation matrix*, $\mathbf{A}$ as shown in Figure 2; or, mathematically as in equation 2.

$$\mathcal{E} = \boldsymbol{\Psi}\mathbf{A} \quad (2)$$

where the columns of $\boldsymbol{\Psi} \in \mathbb{R}^{M \times k}$ represent the M-dimensional embeddings for each speaker in the audio; k is the

---
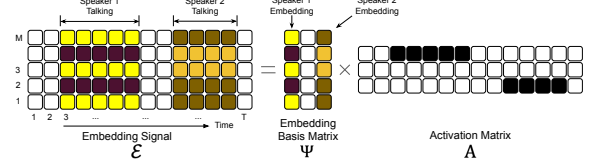[2]A.k.a. YAMNet: https://tfhub.dev/google/yamnet/1

Figure 2: *Model of the Embedding Signal as a matrix product of Embedding Basis Matrix and Activation Matrix*

maximum number of allowed speakers, and T is the length of the embedding signal, i.e. the number of timesteps. The rows of $\mathbf{A} \in \mathbb{R}^{k \times T}$ represent which speaker is activated at a certain timestep. The elements of $\mathbf{A}$ can take a value in 0 to 1. This model converts the diarization problem into a matrix factorization problem. This formulation does not require us to know the *exact number* of the speakers in the audio. As long as the value of k is large enough, this formulation works by setting the activation and embedding for all the unused speakers to zero.

2.2.1. Formulating the optimization problem

We solve the matrix factorization problem using a sparse optimization approach. We minimize the ℓ_1-norm[3] of the difference, $\|\mathcal{E} - \boldsymbol{\Psi}\mathbf{A}\|_1$, to enforce model constraint as in equation 2. In order to obtain the unique solution of this under-determined problem, we enforce sparsity constraint over both $\boldsymbol{\Psi}$ and $\mathbf{A}$ by minimizing their ℓ_1 norms, respectively. From the diarization perspective, the sparsity constraint over $\boldsymbol{\Psi}$ enforces that the embedding signal be reconstructed by utilizing as few speakers as possible. On the other hand, sparsity over $\mathbf{A}$ enforces the solution contains as few nonzero elements for $\mathbf{A}$ as possible. Furthermore, we also enforce that columns of $\boldsymbol{\Psi}$ are each within a unit disk and the elements of $\mathbf{A}$ is within the range $[0, 1]$.

The exact values of the embeddings may vary slightly from time to time depending on the background music or noise in the audio. To address this concern, we introduce a *jitter loss* that enforces continuity and penalizes producing too many broken values in the rows of $\mathbf{A}$. This loss is expressed as an ℓ_1-norm of the difference between consecutive values in the rows of $\mathbf{A}$. The overall optimization problem is shown below:

$$\underset{\text{w.r.t } \boldsymbol{\Psi}, \mathbf{A}}{\text{minimize}} \quad \|\mathcal{E} - \boldsymbol{\Psi}\mathbf{A}\|_1 + \lambda_1\|\boldsymbol{\Psi}\|_1 + \lambda_2\|\mathbf{A}\|_1 + \lambda_3\mathbf{J} \quad (3)$$

$$\text{subject to,} \quad \forall_{r,t} \ 0.0 \leq \mathbf{A}[r,t] \leq 1.0$$
$$\text{and,} \quad \forall_t \|\boldsymbol{\Psi}[:,t]\|_2 \leq 1.0$$

$$\text{where, jitter loss,} \quad \mathbf{J} = \frac{1}{kT}\sum_{r=1}^{k}\sum_{t=2}^{T}|\mathbf{A}[r,t] - \mathbf{A}[r,t-1]|$$

2.2.2. Solving the optimization problem

The objective function shown in equation 3 is non-convex. However, when either $\boldsymbol{\Psi}$ or $\mathbf{A}$ is held fixed, it becomes convex over the other parameter. We solve this optimization problem by alternatingly updating the two model parameters. We utilize the *fast iterative shrinkage thresholding algorithm* [28] (FISTA) to enforce sparsity over the model parameters, $\boldsymbol{\Psi}$ and $\mathbf{A}$. In addition, we project the parameters in their respective feasibility space at every iteration. We implement the optimization algorithm in Tensorflow® to take advantage of the automatic

---
[3]The ℓ_1-norm of $\mathbf{X} \in \mathbb{R}^{U,V}$, $\|\mathbf{X}\|_1 := \sum_{u=1}^{U}\sum_{v=1}^{V}|\mathbf{X}[u,v]|$

Algorithm 1: Speaker diarization

Input: $\mathcal{E}, k$
Output: $\mathbf{\Psi}, \mathbf{A}$

1 $i \leftarrow 0$;
2 $\mathbf{A} \leftarrow$ random, $\mathbf{\Psi} \leftarrow$ random;
3 **while** *not converge* **do**
4 $\quad$ Compute loss,
$\quad\quad \mathbf{L} = \|\mathcal{E} - \mathbf{\Psi A}\|_1 + \lambda_1\|\mathbf{\Psi}\|_1 + \lambda_2\|\mathbf{A}\|_1 + \lambda_3\mathbf{J}$;
5 $\quad$ Calculate gradient of $\mathbf{L}$ w.r.t $\mathbf{\Psi}$, $\nabla_{\mathbf{\Psi}}\mathbf{L}$;
6 $\quad$ Update $\mathbf{\Psi}$ using Adam:
$\quad\quad \mathbf{\Psi}^{(i+1)} \leftarrow \mathbf{\Psi}^{(i)} - \gamma_{\mathbf{\Psi}} \nabla_{\mathbf{\Psi}}\mathbf{L}$;
7 $\quad \mathbf{\Psi}^{(i+1)} \leftarrow \mathbf{shrink}(\mathbf{\Psi}^{(i+1)})$;
8 $\quad \mathbf{\Psi}^{(i+1)} \leftarrow \mathbf{project}_{\mathbf{unitdisk}}(\mathbf{\Psi}^{(i+1)})$;
9 $\quad$ Recompute loss,
$\quad\quad \mathbf{L} = \|\mathcal{E} - \mathbf{\Psi A}\|_1 + \lambda_1\|\mathbf{\Psi}\|_1 + \lambda_2\|\mathbf{A}\|_1 + \lambda_3\mathbf{J}$;
10 $\quad$ Calculate gradient of $\mathbf{L}$ w.r.t $\mathbf{A}$, $\nabla_{\mathbf{A}}\mathbf{L}$;
11 $\quad$ Update $\mathbf{A}$ using Adam
$\quad\quad \mathbf{A}^{(i+1)} \leftarrow \mathbf{A}^{(i)} - \gamma_A \nabla_{\mathbf{A}}\mathbf{L}$;
12 $\quad \mathbf{A}^{(i+1)} \leftarrow \mathbf{shrink}(\mathbf{A}^{(i+1)})$;
13 $\quad \mathbf{A}^{(i+1)} \leftarrow \mathbf{project}_{[\mathbf{0,1}]}(\mathbf{A}^{(i+1)})$;
14 $\quad i \leftarrow i + 1$

---

gradient computation using the Wengert [29] list[4] approach. As a result, we do not need to analytically compute the gradients of the objective function. Similar functionality is also available in PyTorch®. We use Adam [30] for updating the model parameters. The overall optimization process is shown in Algorithm 1.

The **shrink** operation on a matrix $\mathbf{X}$ is defined in equation 4 where γ and λ represents the learning rate and the Lagrange multiplier respectively. This operation pushes each component of the matrix towards zero and thus achieves sparsity quickly [28].

$$\mathbf{shrink}(\mathbf{X}) := sign(\mathbf{X})max(0, |\mathbf{X}| - \gamma\lambda) \quad (4)$$

The **project$_{\mathbf{unitdisk}}$** and **project$_{[\mathbf{0,1}]}$** are two projection operations responsible for keeping the magnitudes of the model parameters in check. They are defined in equation 5 and equation 6, respectively.

$$\mathbf{project}_{\mathbf{unitdisk}}(\mathbf{X}) := \frac{\mathbf{X}[:,c]}{\|\mathbf{X}[:,c]\|_2} \quad \forall_{\text{column index},c} \quad (5)$$

$$\mathbf{project}_{[\mathbf{0,1}]}(\mathbf{X}) := \max\{0, \min(1, \mathbf{X})\} \quad (6)$$

For λ_1, λ_2, and λ_3, we use the values 0.3366, 0.2424, and 0.06 respectively. We obtained these values using a Bayesian Hyper-parameter search as implemented in the Google Cloud Platform [31]. We used a randomly sampled subset (n=60) from the validation set.

2.2.3. Determining the maximum number of speakers

The use of ℓ_1 norm of $\mathbf{\Psi}$ in equation 3 ensures that algorithm 1 utilizes as few embedding vectors as possible to reconstruct the embedding signal, $\mathcal{E}$. This process relieves the users from the burden of supplying an *exact* number of speakers to the algorithm. However, setting a reasonable *maximum* number of speakers (k) is still crucial because setting it too high would

---
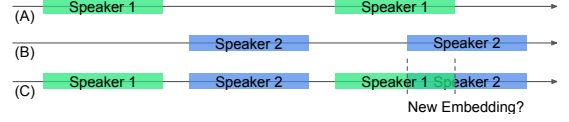[4]A.k.a. *Gradient Tape* in Tensorflow

Figure 3: *A practical situation where the proposed algorithm is able to recover the true speaking activity from an overlapping region*

make the computation unnecessarily slow. We obtain a reasonable value for k from the estimated rank of the embedding signal. The columns of the embedding signal can be thought of just the embeddings of different speakers that are "copied" over time. In that scenario, it is possible to compute the number of the speakers by computing the *Singular Value Decomposition* of the embedding signal and counting the number of non-zero singular values. However, the measurement noise in the speaker embeddings makes the situation a bit complicated because it yields many small but non-zero singular values. We circumvent this situation by sorting the singular values in a descending order and find the "knee" of the curve using the *Kneedle* [32] algorithm. We multiply the location of the knee by a factor of 2.5 which gives us a useful margin of error for the upper bound for number of speakers and ensures that the sparsity constraint for the embedding basis matrix still holds.

2.2.4. Overlap awareness

The utilization of the sparsity constraint along with the linearity property as in equation 1 allows the proposed algorithm to practically identify and disambiguate regions with overlapping speakers. We describe the rationale for this capability with the help of figure 3. Let us assume the rows *A* and *B* in figure 3 show the regions in the audio recording where speaker 1 and speaker 2 are speaking respectively. The resulting embedding signal is shown in row *C*. Noticeably, there is a region in this row where the two speakers overlap. Therefore, as per equation 1, the resulting embedding sequence in this overlapping region will be a linear combination of the embeddings from speaker 1 and speaker 2.

There are two ways of accommodating this resulting embedding sequence using the embedding basis matrix and activity matrix. The overlapping region can be interpreted as a new speaker embedding with the corresponding activity value of 1.0. Alternatively, the activity value for both speakers can be set to be nonzero without introducing a new speaker embedding. Since the introduction of a new embedding would incur additional loss, the optimization algorithm will prefer the second approach. There are settings in which this algorithm will fail to identify overlapping regions (e.g. when a speaker always is overlapped, or when the resulting linear combination of speaker embeddings happens to exactly match the embedding of another speaker), but these are unlikely to occur in practice.

3. Experiments

In order to ground the diarization performance of the proposed algorithm, we evaluate it against a few publicly available speaker diarization solutions. We conduct the evaluation over the two largest publicly available annotated datasets: *This American Life Podcast Dataset* [33] and *VoxConverse* [34].

The "This American Life Podcast" dataset [33] consists of 663 episodes from a podcast with the same name. The dataset

Table 1: *Diarization results on two data sets*

Method	VoxConverse data set				This American Life Podcast			
	DER	Purity	Coverage	F	DER	Purity	Coverage	F
GCP	0.39	0.74	0.92	0.80	0.63	0.47	0.93	0.62
Spectral Clust.	0.50	0.70	0.87	0.76	0.53	0.73	0.77	0.73
PyAnnote v1.1	0.42	0.68	0.92	0.76	0.38	0.73	0.75	0.74
PyAnnote v2.0	0.12	0.90	0.92	0.91	0.25	0.94	0.92	0.93
One Speaker	0.53	0.57	1.00	0.70	0.77	0.32	1.00	0.47
Random	0.67	0.57	0.52	0.52	0.83	0.32	0.48	0.37
Sparse Opt. (Ours)	0.56	0.90	0.60	0.70	0.35	0.84	0.84	0.83

consists of dialog transcripts and human annotated speaker labels at the utterance level. It consists of a total of 637 hours of audio. On average, each audio recording is 57.7 minutes long and contains 18 speakers in this dataset.

The "VoxConverse" dataset [34] consists of 448 audio recordings from various speakers. This dataset was created by mixing together a collection of audio samples from YouTube videos. The speaker labels were obtained using an automated pipeline that can recognize the speakers' face, and can associate the audio with the speaker by analyzing the facial movements [34]. The average audio length in VoxConverse is 8.5 minutes and contains 5.5 speakers per audio.

We compare the proposed diarization algorithm over several other approaches as shown in Table 1. **GCP** is a commercial baseline representing the speaker diarization service provided by Google Cloud Platform as of March 8th, 2022. **Pyannote** is a Python open-source toolkit which also provides trained diarization models [35]. PyAnnote is a neural-network-based supervised approach for diarization currently released as version 1.1. A successor, PyAnnote v2.0, is currently actively under development with the technical details undisclosed at the time of writing. **Spectral Clustering** is an in-house implementation of the classical diarization pipeline using the VggVox [26] embedding and spectral clustering. Comparison against this baseline provides an indication of how much of the change in the evaluation metrics is caused by the sparse optimization algorithm alone, and not by the improved discriminative capability of the embeddings. We also define two naive baselines to evaluate the lowest bar for diarization: **One Speaker**, where every word in the transcript is assigned to a single speaker, and **Random**, where every word in the transcript is randomly assigned to some speaker.

For evaluation, we use some standard diarization performance metrics from the Pyannote metrics [35] python library. **Diarization Error Rate (DER)** is the sum of the duration of non-speech regions incorrectly classified as speech (*false alarm*), the duration of speech regions incorrectly classified as non-speech (*missed detection*), and the duration of speaker *confusion*, as a ratio of the total duration of speech for all speakers.

Purity is a precision-related measure representing the quality of the each *predicted* speech segments. It is represented by equation 7, where |cluster| and |speaker| represent the speech duration of the predicted speech segments and the reference speech segments respectively. |cluster ∩ speaker| represents the duration of their intersection. **Coverage** is the corresponding recall-related measure calculated for each *reference* speech segment as in equation 8. The **F** score is the harmonic mean of purity and coverage. A better system has lower DER, or higher purity, coverage, or F score.

$$\text{purity} = \frac{\sum_{\text{cluster}} \max_{\text{speaker}} |\text{cluster} \cap \text{speaker}|}{\sum_{\text{cluster}} |\text{cluster}|} \quad (7)$$

$$\text{coverage} = \frac{\sum_{\text{speaker}} \max_{\text{cluster}} |\text{speaker} \cap \text{cluster}|}{\sum_{\text{speaker}} |\text{speaker}|} \quad (8)$$

It is evident from Table 1 that the sparse optimization algorithm works much better in the This American Life podcast dataset than the VoxConverse dataset. This behavior is due to the fact that VoxConverse audio recordings are much shorter and, therefore, result in fewer columns in the embedding signal as compared to the rows. This transforms the matrix factorization problem from the domain of *under-determined* systems to an *over-determined* system that violates the assumptions for sparse optimization formulation. However, we are interested in diarization solutions for the podcast applications—which are typically long, and therefore suitable for our use case.

The overall diarization purity for the proposed algorithm is better than GCP, spectral clustering, and PyAnnote v1.1 by a large margin: the predicted clusters are more pure, i.e. not a mix of speakers, at a certain cost in coverage. The overall performance (F-score) also shows similar trend. It is second only to PyAnnote v2.0—for which the algorithm is still undisclosed. If PyAnnote v2.0 works by taking a supervised approach it has a risk of being susceptible to domain mismatch. In addition, supervised approaches are often difficult to maintain due to difficulties in collecting data and the amount of engineering effort it takes to keep the dataset balanced for preventing bias and ensuring fairness.

4. Conclusions

We present a novel method for speaker diarization that is unsupervised—does not require human annotations which are difficult to collect and maintain. The algorithm performance beats a commercial solution (GCP) across all standard metrics. It is completely audio-based, therefore, agnostic to transcription or any other language-dependent processing. In addition, it removes the burden of supplying an exact number of speakers from the users; and thus works in a tune-free or self-tuned fashion. Accommodating for the overlapping speech is built into the design of the estimator, therefore removes the necessity for any ad-hoc post-processing. Being a first order optimization approach that is implemented in TensorFlow, this algorithm is massively parallelizable and has great potential for speed improvement on the right kind of processing infrastructure (e.g. utilizing TPU's or GPU's).

5. References

[1] R. Jones, B. Carterette, A. Clifton, M. Eskevich, G. J. Jones, J. Karlgren, A. Pappu, S. Reddy, and Y. Yu, "TREC 2020 Podcasts Track Overview," in *The Twenty-Ninth Text REtrieval Conference Proceedings (TREC 2020)*. NIST, 2021.

[2] R. Jones, H. Zamani, M. Schedl, C.-W. Chen, S. Reddy, A. Clifton, J. Karlgren, H. Hashemi, A. Pappu, Z. Nazari, L. Yang, O. Semerci, H. Bouchard, and B. Carterette, "Current Challenges and Future Directions in Podcast Information Access," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[3] A. Clifton, S. Reddy, Y. Yu, A. Pappu, R. Rezapour, H. Bonab, M. Eskevich, G. J. F. Jones, J. Karlgren, B. Carterette, and R. Jones, "100,000 Podcasts: A Spoken English Document Corpus," in *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 2020.

[4] J. Karlgren, R. Jones, B. Carterette, A. Clifton, M. Eskevich, G. J. F. Jones, S. Reddy, E. Tanaka, and M. I. Tanveer, "TREC 2021 Podcasts Track Overview," in *The Thirtieth Text REtrieval Conference Proceedings (TREC 2021)*. NIST, 2022.

[5] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A review of speaker diarization: Recent advances with deep learning," *Computer Speech & Language*, 2022.

[6] 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*, 2010.

[7] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

[8] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[9] H. Ning, M. Liu, H. Tang, and T. S. Huang, "A spectral clustering approach to speaker diarization," in *Ninth International Conference on Spoken Language Processing*, 2006.

[10] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," *IEEE Transactions on Audio, Speech, and Language Processing*, 2013.

[11] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker diarization with lstm," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[12] G. Sell, D. Snyder, A. McCree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe *et al.*, "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge." in *19th Annual Conference of the International Speech Communication Association (Interspeech)*, 2018.

[13] P. Kenny, D. Reynolds, and F. Castaldo, "Diarization of telephone conversations using factor analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, 2010.

[14] F. Valente, P. Motlicek, and D. Vijayasenan, "Variational Bayesian speaker diarization of meeting recordings," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2010.

[15] L. Bullock, H. Bredin, and L. P. Garcia-Perera, "Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[16] Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with self-attention," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019.

[17] S. Dey, S. R. Madikeri, and P. Motlicek, "End-to-end text-dependent speaker verification using novel distance measures." in *19th Annual Conference of the International Speech Communication Association (Interspeech)*, 2018.

[18] L. E. Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," *arXiv preprint arXiv:1907.05337*, 2019.

[19] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland, "Overlapped speech detection for improved speaker diarization in multiparty meetings," in *2008 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2008.

[20] J. T. Geiger, F. Eyben, B. Schuller, and G. Rigoll, "Detecting overlapping speech with long short-term memory recurrent neural networks," in *14th Annual Conference of the International Speech Communication Association (Interspeech)*, 2013.

[21] D. Raj, L. P. Garcia-Perera, Z. Huang, S. Watanabe, D. Povey, A. Stolcke, and S. Khudanpur, "Dover-lap: A method for combining overlap-aware diarization outputs," in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021.

[22] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[23] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus mathematique*, vol. 346, 2008.

[24] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, 2006.

[25] R. Vipperla, J. T. Geiger, S. Bozonnet, D. Wang, N. Evans, B. Schuller, and G. Rigoll, "Speech overlap detection and attribution using convolutive non-negative sparse coding," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4181–4184.

[26] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, 2020.

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, 2009.

[29] R. E. Wengert, "A simple automatic derivative evaluation program," *Communications of the ACM*, vol. 7, no. 8, 1964.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1487–1495.

[32] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *31st international conference on distributed computing systems workshops*. IEEE, 2011.

[33] H. H. Mao, S. Li, J. McAuley, and G. Cottrell, "Speech recognition and multi-speaker diarization of long conversations," *arXiv preprint arXiv:2005.08072*, 2020.

[34] J. S. Chung, J. Huh, A. Nagrani, T. Afouras, and A. Zisserman, "Spot the conversation: speaker diarisation in the wild," *arXiv preprint arXiv:2007.01216*, 2020.

[35] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "Pyannote.Audio: Neural building blocks for speaker diarization," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.