TURN-TO-DIARIZE: ONLINE SPEAKER DIARIZATION CONSTRAINED BY TRANSFORMER TRANSDUCER SPEAKER TURN DETECTION

*Wei Xia** *Han Lu** *Quan Wang** *Anshuman Tripathi* *Ignacio Lopez Moreno* *Hasim Sak*

Google LLC, USA

{ ericwxia, luha, quanw, anshumant, elnota, hasim } @google.com

ABSTRACT

In this paper, we present a novel speaker diarization system for streaming on-device applications. In this system, we use a transformer transducer to detect the speaker turns, represent each speaker turn by a speaker embedding, then cluster these embeddings with constraints from the detected speaker turns. Compared with conventional clustering-based diarization systems, our system largely reduces the computational cost of clustering due to the sparsity of speaker turns. Unlike other supervised speaker diarization systems which require annotations of time-stamped speaker labels for training, our system only requires including speaker turn tokens during the transcribing process, which largely reduces the human efforts involved in data collection.

Index Terms— Speaker diarization, speaker turn detection, constrained spectral clustering, transformer transducer

1. INTRODUCTION

Speaker segmentation is a key component in most modern speaker diarization systems [1]. The outputs of speaker segmentation are usually short segments which can be assumed to consist of individual speakers. With these homogeneous segments, we can extract speaker embedding such as i-vector [2] or d-vector/x-vector [3, 4, 5] from each segment to represent its speaker identify. The speaker embeddings can be either directly clustered with conventional clustering algorithms such as K-means [6] or spectral clustering [7], or fed into a supervised model such as unbounded interleaved-state recurrent neural networks (UIS-RNN) [8], discriminative neural clustering (DNC) [9], or permutation-invariant training [10, 11].

There are typically three approaches to the speaker segmentation problem:

1. Uniform speaker segmentation: The entire utterance is divided into segments of uniform length. Although this approach is simple and easy to implement, it is difficult to find a good segment length — long segments may very likely contain speaker turn boundaries, while short segments carry insufficient speaker information. For example, the systems described in [7, 8] are based on segments of a fixed length of 400ms, while the system described in [12] is based on segments of 2s.
2. ASR-based word segmentation: Automatic speech recognition (ASR) models generate word boundaries, which could be used as word-level segmentation. Although we can usually safely assume that a word-level segment comes from a single speaker, word segments are still too short to carry sufficient speaker information.
3. Supervised speaker turn detection (*a.k.a.* speaker change detection): A dedicated model is trained to detect the exact timestamps of speaker turns, such as the systems described in [13, 14].

Apparently, among the above three approaches, supervised speaker turn detection has multiple advantages. First, since a segment covers a full continuous speaker turn, it carries sufficient information to extract robust speaker embeddings. Besides, for very long conversational speech, the number of speaker turns is usually much smaller than the number of appropriate fixed-length short segments — this would largely reduce the computational cost of clustering the segment-wise embeddings.

The speaker turn detection models described in [13, 14] are purely based on acoustic information of the training utterances. These kinds of models fail to leverage the rich semantic information in the data. For example, by only looking at the text transcript of the conversation "How are you I'm good", we can confidently conjecture there is a speaker change between "How are you" and "I'm good".

Although many recent end-to-end speaker diarization systems have shown very promising results [10, 15, 11], these systems usually require a large amount of carefully annotated training data. The annotation process usually requires the human annotator to assign accurate *timestamps* to the speaker turns, and manually identify different speakers across these turns. Our internal study shows that this kind of annotation process takes roughly 2 hours for a single annotator to annotate 10 minutes of audio for one pass.

In this paper, we propose Transformer Transducer [16, 17, 18] based speaker turn detection that is jointly trained with the ASR model [19]. We use a special token `<st>` to represent the speaker turn, and inject this token into ground truth transcripts of the ASR training data. This approach not only makes better use of semantic information in the speech data, but also reduces annotation costs — annotating the `<st>` token as part of transcribing the speech data is much easier than annotating the exact timestamp of the speaker turn events.

Our work shares similarities with the system proposed in [20]. In [20], two role-specific tokens `<spk:dr>` and `<spk:pt>` are injected to the ASR transcripts to indicate speech from the doctor and the patient. However, such a system cannot be used for generic speaker diarization problems where: (1) There could be more than 2 speakers; (2) The speakers are not constrained to specific roles.

The original contributions of this paper include: (1) We proposed an efficient speaker diarization system for streaming on-device applications that do not rely on expensive timestamped annotations; (2) We proposed a transformer transducer-based model for joint ASR and speaker turn detection; (3) We proposed a constrained spectral clustering algorithm that incorporates the prior information from speaker turns into the spectral clustering process.

* Equal contribution. Wei performed this work as an intern at Google.

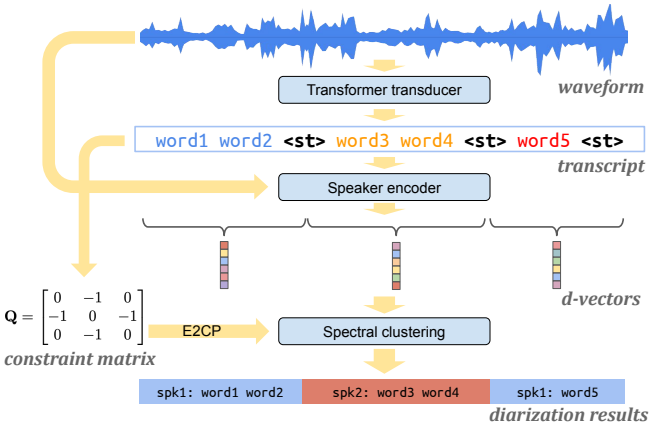

Fig. 1. System architecture of the speaker diarization system.

Table 1. Hyper-parameters of a Transformer block.

Input feature projection	256
Dense layer 1	1024
Dense layer 2	256
Number attention heads	8
Head dimension	64
Dropout ratio	0.1

2. METHODS

2.1. System architecture

The system architecture is shown in Fig. 1. The input utterance is first fed into a transformer transducer model for joint ASR and speaker turn detection. Then the utterance is segmented into speaker turns, and each turn is fed into an LSTM based speaker encoder to extract a d-vector embedding. We use a spectral clustering algorithm to cluster these turn-wise d-vectors, but with constraints from the detected speaker turns[1].

2.2. Online diarization

Since both the transformer transducer-based speaker turn detection model and the LSTM based speaker encoder model are streaming models, the bottleneck of latency is the clustering algorithm. Previous studies have shown that online clustering algorithms such as Links [21] are significantly worse than offline clustering algorithms such as spectral clustering [7]. To have both great performance and low latency at the same time, we use spectral clustering in an online fashion: every time when we have a new speaker embedding, we run spectral clustering on the entire sequence of all existing embeddings. Because speaker embeddings are extracted from speaker turns which are usually sparse, the sequence is usually relatively short even for hour-long conversations, thus making the clustering inexpensive to run and feasible for on-device deployment.

[1]We open sourced the constrained spectral clustering algorithm at https://github.com/wq2012/SpectralCluster

2.3. Transformer transducer based speaker turn detection

Recurrent neural network transducer (RNN-T) [22] is an ASR model architecture that can be trained end-to-end with RNN-T loss. Such architecture includes an audio encoder, a label encoder, and a joint network that produces the final output distribution over all possible labels. We adopt Transformer Transducer (T-T) [16], a variant of the RNN-T model, as the speaker turn detection model for its advantages of faster inference speed, and no long-form deletion issues. We also use a bigram label encoder proposed in [17] to further speed up the decoding via an embedding table lookup.

To create training targets, inspired by [20] that adds speaker roles as part of the transcript (*e.g.* "hello how are you <spk:dr> I am good <spk:pt>"), we add a special speaker turn token <st> between two different speakers' transcripts (*e.g.* "hello how are you <st> I am good <st>") to model speaker turns during training. Compared to audio-only models [13, 14], this model can potentially utilize the language semantics as a signal for speaker segmentation. T-T is trained in a sequence-to-sequence fashion, where the source is log-Mel filterbank energy features, and the target is the transcript that includes both transcript texts and the special speaker turn tokens. At inference time, we ignore all the texts output by the model except for the <st> tokens, and their corresponding timestamps. These timestamps are later used as the speaker boundaries in the diarization system.

We train the T-T model using Fisher [23], the training subset of Callhome American English [24], and an internal dataset collected from around 7500 hours of YouTube videos. Training utterances are segmented into 15 seconds segments with speaker turn tokens added so it fits into the memory more easily. The audio encoder has 15 layers of Transformer blocks. Each block has 32 left context and no right context. The hyper-parameters for each repeated block can be found in Table 1. We also use a stacking layer after the second transformer block to change the frame rate from 30ms to 90ms, and an unstacking layer after the 13th transformer block to change the frame rate from 90ms back to 30ms, to speed up the audio encoder as proposed in [17]. The bi-gram label encoder embedding table has embeddings with a dimension of 256. For the joint network, we have a projection layer that projects the audio encoder output to 256-d. At the output of the joint network, it produces a distribution over 75 possible graphemes with a softmax layer. For optimization, we follow the same hyper-parameters described in [16].

2.4. Speaker encoder

Our speaker encoder is a text-independent speaker recognition model trained with the generalized end-to-end extended-set softmax loss [3, 25]. The speaker encoder model has 3 LSTM layers each with 768 nodes and a projection size of 256. The output of the last LSTM layer is then linearly transformed to the final 256-dimension d-vector. The same model was used in [26].

At inference time, we use the detected speaker turns as signals to reset the LSTM states of the speaker encoder, such that it does not carry information across different turns. For each speaker turn, we use the embedding at roughly 75% of this turn to represent this speaker turn, such that it has sufficient information from this turn, and is not too close to the speaker boundary which could be inaccurate or contain overlapped speech. Besides, because speaker turn detection may have false rejections, to reduce the risk, we further segment any turns that are longer than 6 seconds. This type of segmentation is also used to construct "Must-Link" constraints as described in Sec. 2.5.3.

2.5. Spectral clustering

2.5.1. Recap of spectral clustering

We use the spectral clustering method [27] to predict speaker labels on turn-wise speaker embeddings. Given a set of N data samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, we can construct a similarity graph by computing pairwise similarities a_{ij}. Let $\mathbf{A} \in \mathbb{R}^{N\times N}$ be the affinity matrix of the graph, the affinity of two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is $a_{ij} = \frac{1}{2}(1 + \cos(\mathbf{x}_i, \mathbf{x}_j))$. Since spectral clustering is sensitive to the quality and noise of a similarity graph, we define the following refinement operations [7] on the affinity matrix to model the local neighborhood relationships between data samples.

1. Row-wise soft-thresholding with p-percentile: affinity values that are larger than the p-percentile of the row are binarized to 1; affinity values that are smaller than the p-percentile are multiplied with 0.01.
2. We apply an average symmetrization operation to make the affinity matrix symmetric: $\hat{\mathbf{A}} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^\mathsf{T})$

Gaussian Blur operation [7] is also applied at the beginning for window-wise dense embeddings to smooth and denoise the data. We find that the diarization performance is significantly affected by the hyper-parameter p for the p-percentile. The default p is 0.95 in our experiments. As proposed in [28], we can use a ratio value $r(p)$ as a good proxy of the Diarization Error Rate (DER). The ratio $r(p) = \frac{\sqrt{1-p}}{g_p}$, where g_p is the normalized maximum eigen gap. By minimizing this ratio $r(p)$, we can automatically select an appropriate p-percentile for our spectral clustering algorithm without a holdout development set. Note that for every search step, the auto-tuning method requires an eigen-decomposition operation of the affinity matrix, which is the major bottleneck of the clustering algorithm. If the speaker embeddings are dense, the size of the affinity matrix and therefore the computational cost become very large.

Next we define a graph Laplacian matrix $\mathbf{L}$. Given an affinity matrix $\mathbf{A}$, the degree matrix $\mathbf{D}$ is a diagonal matrix where the diagonal elements $d_{ii} = \sum_{j=1}^{N} a_{ij}$. The unnormalized Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, and the normalized Laplacian $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. In our experiments, we use the normalized graph Laplacian $\bar{\mathbf{L}}$. To perform spectral clustering,

- We apply eigen-decomposition and estimate the speaker number k using the maximum eigengap method.
- We choose the first k eigen-vectors and apply a row-wise renormalization of the spectral embeddings. K-means algorithm is applied on the spectral embeddings to predict class labels.

2.5.2. Speaker turn priors

Given a sequence of speaker segments and speaker turn information, we know that two *neighboring* segments adjacent to the <st> token are from different speakers. Therefore, speaker turn prior information can be used as pairwise constraints to guide the clustering process. Pairwise constraints, unlike the class labels of data, do not provide explicit class information and are considered a weaker form of supervisory information. Using the proposed T-T speaker turn detection in Sec. 2.3, we can predict a confidence score for each speaker turn token <st>. The general objective is to encourage speaker labels of segments across high confidence <st> to be different and speaker labels of segments without <st> token or across very low confidence <st> to be the same.

2.5.3. Spectral clustering with pairwise speaker turn constraints

With the pairwise constraints from speaker turn side information, we can perform a constrained spectral clustering that tries to find a partition (or multiple partitions) that maximizes constraint satisfaction and minimizes the cost on the similarity graph.

Let $\mathbf{Q} \in \mathbb{R}^{N\times N}$ be a constraint matrix. If there is a speaker turn between segment i and $i+1$, and the confidence of the <st> token $c(\texttt{<st>})$ is larger than a threshold σ, we define this pair as a "Cannot-link" (CL) [29]. If there is no speaker turn between two segments, we define it as a "Must-Link" (ML). $\mathbf{Q}_{i,j} = 0$ if i, j are not neighboring segments.

$$\mathbf{Q}_{i,j} = \begin{cases} -1, & \text{If } (i,j) \in \text{CL and } c(\texttt{<st>}) > \sigma; \\ +1, & \text{If } (i,j) \in \text{ML}; \\ 0, & \text{Otherwise.} \end{cases} \tag{1}$$

The generated constraint matrix is a banded sparse matrix since we only have limited speaker turns. To fully utilize the inherent information from the speaker turns, we can infer more constraint information using the Exhaustive and Efficient Constraint Propagation (E2CP) method in [30].

First, we divide the pairwise constraint propagation problem into a two-class label propagation sub-problem, where we treat an ML pair as a positive class and a CL pair as a negative class. The class labels are propagated in vertical and horizontal directions respectively. Let $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, which is the symmetric regularization of the unrefined affinity $\mathbf{A}$. $\mathbf{Z}$ is the initial constraint matrix defined in Eq. 1. A parameter α is used to control the relative amount of constraint information from its neighbors and the initial constraints. It is set to 0.4 in our experiments. We perform vertical propagation first until the convergence and then the horizontal propagation. By combining these two propagations, we diffuse the pairwise constraints to the whole graph. With the E2CP algorithm, the final propagated constraint matrix $\mathbf{Q}^*$ has a closed-form feasible solution, which is formulated as below,

$$\mathbf{Q}^* = (1-\alpha)^2(I - \alpha\bar{\mathbf{A}})^{-1}\mathbf{Z}(I - \alpha\bar{\mathbf{A}})^{-1}. \tag{2}$$

Using this propagated constraint matrix $\mathbf{Q}^*$, we can obtain an adjusted affinity matrix $\hat{\mathbf{A}}$, where

$$\hat{\mathbf{A}}_{ij} = \begin{cases} 1 - \left(1 - \mathbf{Q}^*_{ij}\right)\left(1 - \mathbf{A}_{ij}\right), & \text{If } \mathbf{Q}^*_{ij} \geq 0\,; \\ \left(1 + \mathbf{Q}^*_{ij}\right)\mathbf{A}_{ij}, & \text{If } \mathbf{Q}^*_{ij} < 0\,. \end{cases} \tag{3}$$

For constraint $\mathbf{Q}_{ij} > 0$, it increases the similarity between the sample $\mathbf{x}_i$ and $\mathbf{x}_j$; if the constraint is negative, the similarity is decreased. After this operation, we still perform the normalized Laplacian matrix based spectral clustering to predict cluster labels[2].

3. EXPERIMENTS

3.1. Data and metrics

The training data for the speaker turn detection model and the speaker encoder model have been described in Sec. 2.3 and Sec. 2.4, respectively. To evaluate our speaker diarization system, we use a vendor-provided call center domain dataset. This dataset consists of anonymized utterances containing telephone conversations between call center attendants and customers, and can be divided into two subsets:

[2]To summarize, with E2CP, the workflow is: affinity → constraint → refinement → Laplacian.

Table 2. Confusion (%), total DER (%) and GFLOPS/s on three datasets for different embeddings and methods.

System	Method	Inbound		Outbound		Callhome Eval		GFLOP/s	GFLOP/s
		Conf.	DER	Conf.	DER	Conf.	DER	at 10min	at 1h
Dense d-vector	Dense	17.98	22.13	10.66	15.97	5.39	7.76	0.85	36.54
	Dense + Auto-tune	14.09	18.24	9.56	14.88	5.42	7.79	4.76	361.37
Turn-to-diarize	Turn	17.87	19.43	8.41	10.34	8.23	10.08	1.00	1.18
	Turn + E2CP	17.21	18.77	7.94	9.86	3.56	5.41	1.00	1.18
	Turn + Auto-tune	13.83	15.39	7.01	8.93	5.11	6.95	1.02	2.81
	Turn + E2CP + Auto-tune	**13.66**	**15.22**	**6.86**	**8.78**	**3.49**	**5.33**	1.02	2.81

1. The "Outbound" subset, which includes 450 conversations initiated by the call center. This dataset has approximately 35 hours of speech in total. Each utterance has 2 speakers.
2. The "Inbound" subset, which includes 250 conversations initiated by customers. This dataset has approximately 22 hours of speech in total. Each utterance has 2 to 10 speakers.

Apart from the internal call center domain dataset, we also evaluate our diarization system on the Callhome American English data (LDC97S42) [24]. The Callhome American English corpus is divided into the train, dev, and eval sets. As the train subset has been used for training the speaker turn detection model, we report the diarization results on the eval set of 20 utterances, which is about 1.7 hours of recordings in total.

We report the Diarization Error Rate (DER) computed with the pyannote.metrics library [31], and follow the same evaluation protocols as [7, 8].

3.2. Experimental results

We use the speaker diarization system described in [7] as our baseline system (using the same speaker encoder model as described in Sec. 2.4), and refer to it as the "dense d-vector" system, as the speaker embeddings are extracted from 400ms short segments. In Table 2, we show the experimental results of the "dense d-vector" and the proposed "turn-to-diarize" systems on the Internal Inbound, Outbound datasets, as well as the publicly available Callhome evaluation set. We report the total Diarization Error Rate (DER) and speaker Confusion Error Rate. The remaining errors are from False Alarm (FA) and Miss which are mostly caused by the Voice Activity Detection. As shown in the table, the turn-to-diarize method achieves better diarization results on the Inbound, and Outbound datasets, compared with the dense d-vector system. There is a relative 12.20% and 35.25% reduction in DER on the Inbound and Outbound datasets respectively. It shows that longer-duration speaker turn embeddings that capture more speaker characteristics might be more useful for diarization.

Moreover, the spectral clustering algorithm relies on the quality of the similarity graph. We find that pruning small and noisy values with the hyper-parameter p-percentile is essential to construct a good graph. It significantly impacts the diarization performance. The auto-tuning method [28] based on the ratio $r(p)$ does not require a holdout development set to tune the hyper-parameters, and we use this method to automatically select a good p-percentile. For all three datasets, the p-percentile search range is from 0.4 to 0.95 with a step size of 0.05. The auto-tuning method is tuned per-utterance and requires one operation of eigen-decomposition at each search step. When we use the dense d-vector method, however, the size of the Laplacian matrix is very high, so the computational cost of the eigen-decomposition operation is much more expensive and it causes much larger latency compared with the turn-to-diarize method.

Another advantage of the turn-to-diarize method is that we can use speaker turn prior information as pairwise constraints to guide the clustering process. For the dense d-vector approach, a single segment may cross the speaker turn boundary. It may contain speech from two speakers and therefore causes more confusion errors. Comparing the "Turn" only and "Turn + E2CP" methods, we can observe a relative 3.40%, 4.64%, and 46.33% reduction of DER on the Inbound, Outbound, and Callhome datasets respectively. It indicates the detected speaker turns are not only useful for segmenting the input, but also helpful to constrain the spectral clustering. Moreover, we notice that the E2CP method usually works better when a good p-percentile is not selected. If the similarity graph is already well-constructed with a good p-percentile, the effect of the E2CP method is marginal.

The auto-tuning method can also be combined with the speaker turn constraints. In Table 2, the combination of E2CP and auto-tune with the turn-to-diarize system achieves the best results on all three datasets. It consistently improves the performance of the "Turn" only method and the best results of the dense system by a large margin, indicating the effectiveness of our proposed methods.

3.3. Computational cost

In Table 2 we also include floating-point operations per second (FLOP/s) analysis for each system after running for 10min and 1h. This analysis assumes: dense d-vector is based on 400ms segments; average speaker turn length is 4s; the average number of speakers is 4; auto-tune searches for 10 values of p; and clustering runs every 4s. As we can see, turn-to-diarize is dominated by the speaker turn detection (578 MFLOP/s) and speaker encoder (415 MFLOP/s) neural networks, and the costs of eigen-decomposition, E2CP, Laplacian and K-Means are almost negligible even after processing 10min of audio. For dense d-vector, the computational cost (mostly eigen-decomposition) significantly increases when the sequence grows and becomes unacceptable.

4. CONCLUSIONS

We proposed a speaker diarization system for streaming on-device applications. The system is based on a transformer transducer model for joint speech recognition and speaker turn detection. The detected speaker turns are not only used to segment the input, but also to constrain the spectral clustering of the turn-wise speaker embeddings. By clustering turn-wise embeddings instead of short segment-wise embeddings, we significantly reduced computational cost, and achieved offline performance with online latency. One future work is to retrain our transformer transducer on multilingual datasets to make our speaker diarization system language independent.

5. REFERENCES

[1] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J Han, Shinji Watanabe, and Shrikanth Narayanan, "A review of speaker diarization: Recent advances with deep learning," *arXiv preprint arXiv:2101.09624*, 2021.

[2] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[3] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *ICASSP*. IEEE, 2018, pp. 4879–4883.

[4] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, vol. 650, 2017.

[5] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-Vectors: Robust dnn embeddings for speaker recognition," in *ICASSP*. IEEE, 2018, pp. 5329–5333.

[6] Stephen H Shum, Najim Dehak, Réda Dehak, and James R Glass, "Unsupervised methods for speaker diarization: An integrated and iterative approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2015–2028, 2013.

[7] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopez Moreno, "Speaker diarization with LSTM," in *ICASSP*. IEEE, 2018, pp. 5239–5243.

[8] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang, "Fully supervised speaker diarization," in *ICASSP*. IEEE, 2019, pp. 6301–6305.

[9] Qiujia Li, Florian L Kreyssig, Chao Zhang, and Philip C Woodland, "Discriminative neural clustering for speaker diarisation," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2021.

[10] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Kenji Nagamatsu, and Shinji Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proc. Interspeech*, 2019, pp. 4300–4304.

[11] Quan Wang, Yash Sheth, Ignacio Lopez Moreno, and Li Wan, "Speaker diarization using an end-to-end model," Google Patents, 2019.

[12] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree, "Speaker diarization using deep neural network embeddings," in *ICASSP*. IEEE, 2017, pp. 4930–4934.

[13] Ruiqing Yin, Hervé Bredin, and Claude Barras, "Speaker change detection in broadcast tv using bidirectional long short-term memory networks," in *Proc. Interspeech*, 2017, pp. 3827–3831.

[14] Ruiqing Yin, Hervé Bredin, and Claude Barras, "Neural speech turn segmentation and affinity propagation for speaker diarization," in *Proc. Interspeech*, 2018, pp. 1393–1397.

[15] Soumi Maiti, Hakan Erdogan, Kevin Wilson, Scott Wisdom, Shinji Watanabe, and John R Hershey, "End-to-end diarization for variable number of speakers with local-global networks and discriminative speaker embeddings," in *ICASSP*. IEEE, 2021, pp. 7183–7187.

[16] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP*. IEEE, 2020, pp. 7829–7833.

[17] Anshuman Tripathi, Jaeyoung Kim, Qian Zhang, Han Lu, and Hasim Sak, "Transformer transducer: One model unifying streaming and non-streaming speech recognition," *arXiv preprint arXiv:2010.03192*, 2020.

[18] Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[19] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*. IEEE, 2019, pp. 6381–6385.

[20] Laurent El Shafey, Hagen Soltau, and Izhak Shafran, "Joint speech recognition and speaker diarization via sequence transduction," in *Proc. Interspeech*, 2019, pp. 396–400.

[21] Philip Andrew Mansfield, Quan Wang, Carlton Downey, Li Wan, and Ignacio Lopez Moreno, "Links: A high-dimensional online clustering method," *arXiv preprint arXiv:1801.10123*, 2018.

[22] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[23] Christopher Cieri, David Miller, and Kevin Walker, "The Fisher corpus: A resource for the next generations of speech-to-text.," in *LREC*, 2004, vol. 4, pp. 69–71.

[24] A Canavan, D Graff, and G Zipperlen, "CALLHOME American English speech LDC97S42," LDC Catalog. Philadelphia: Linguistic Data Consortium, 1997.

[25] Jason Pelecanos, Quan Wang, and Ignacio Lopez Moreno, "Dr-Vectors: Decision residual networks and an improved loss for speaker recognition," in *Proc. Interspeech*, 2021.

[26] Rajeev Rikhye, Quan Wang, Qiao Liang, Yanzhang He, Ding Zhao, Arun Narayanan, Ian McGraw, et al., "Personalized keyphrase detection using speaker and environment information," in *Proc. Interspeech*, 2021.

[27] Ulrike Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[28] Tae Jin Park, Kyu J Han, Manoj Kumar, and Shrikanth Narayanan, "Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap," *IEEE Signal Processing Letters*, vol. 27, pp. 381–385, 2019.

[29] Sugato Basu, Ian Davidson, and Kiri Wagstaff, *Constrained clustering: Advances in algorithms, theory, and applications*, CRC Press, 2008.

[30] Zhiwu Lu and Yuxin Peng, "Exhaustive and efficient constraint propagation: A graph-based learning approach and its applications," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 306–325, 2013.

[31] Hervé Bredin, "`pyannote.metrics`: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Proc. Interspeech*, 2017, pp. 3587–3591.

[32] Rohit Prabhavalkar, Raziel Alvarez, Carolina Parada, Preetum Nakkiran, and Tara N Sainath, "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks," in *ICASSP*. IEEE, 2015, pp. 4704–4708.

[33] Rubén Zazo Candil, Tara N Sainath, Gabor Simko, and Carolina Parada, "Feature learning with raw-waveform cldnns for voice activity detection," in *Proc. Interspeech*, 2016.

[34] Yue Fan, JW Kang, LT Li, KC Li, HL Chen, ST Cheng, PY Zhang, ZY Zhou, YQ Cai, and Dong Wang, "CN-CELEB: a challenging Chinese speaker recognition dataset," in *ICASSP*. IEEE, 2020, pp. 7604–7608.

[35] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett, "Darpa TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," NASA STI/Recon technical report, 1993.

[36] Junichi Yamagishi, Christophe Veaux, Kirsten MacDonald, et al., "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit," 2019.

[37] Richard Lippmann, Edward Martin, and D Paul, "Multi-style training for robust isolated-word speech recognition," in *ICASSP*. IEEE, 1987, vol. 12, pp. 705–708.

[38] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *ICASSP*. IEEE, 2017, pp. 5220–5224.

[39] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara Sainath, and Michiel Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. Interspeech*, 2017.

[40] Rajeev Rikhye, Quan Wang, Qiao Liang, Yanzhang He, and Ian McGraw, "Multi-user voicefilter-lite via attentive speaker embedding," *arXiv preprint arXiv:2107.01201*, 2021.

Appendices

A. FEATURE FRONTEND

We used a shared feature frontend for the speaker turn detection model in Section 2.3 and the speaker encoder model in Section 2.4. This frontend first applies automatic gain control [32] to the input audio, then extracts 32ms-long Hanning-windowed frames with a step of 10ms. For each frame, 128-dimensional log Mel-filterbank energies (LFBE) are computed in the range between 125Hz and 7500Hz. These filterbank energies are then stacked by 4 frames and subsampled by 3 frames, resulting in final features of 512 dimensions with a frame rate of 30ms. These features are then filtered by a CLDNN based Voice Activity Detection (VAD) [33] before fed into the speaker turn detection and the speaker encoder models.

B. ADDITIONAL DETAILS ON SPEAKER ENCODER MODEL

We introduced our LSTM-based speaker encoder model in Section 2.4. The training data of this model include a vendor collected multi-language speech query dataset covering 37 locales, as well as LibriVox, CN-Celeb [34], TIMIT [35], and VCTK [36]. Multi-style training (MTR) [37, 38, 39] with SNR ranging from 3dB to 15dB is applied during the training process for noise robustness. The same speaker encoder model was also used in [26, 40].

C. EXHAUSTIVE AND EFFICIENT CONSTRAINT PROPAGATION APPROACH

The Exhaustive and Efficient Constraint Propagation (E2CP) [30] method is described in Algorithm 1.

Algorithm 1 Exhaustive and Efficient Constraint Propagation (E2CP) method

Require: Initial constraint matrix $\mathbf{Z} = \mathbf{Q}(0)$, matrix $\bar{\mathbf{A}}$, propagation parameter α.
  while $\mathbf{Q}_v(t)$ not converged to $\mathbf{Q}_v^*$ **do**
    $\mathbf{Q}_v(t+1) = \alpha\bar{\mathbf{A}}\mathbf{Q}_v(t) + (1-\alpha)\mathbf{Z}$ ▷ Vertical propagation
  end while
  while $\mathbf{Q}_h(t)$ not converged to $\mathbf{Q}_h^*$ **do**
    $\mathbf{Q}_h(t+1) = \alpha\mathbf{Q}_h(t)\bar{\mathbf{A}} + (1-\alpha)\mathbf{Q}_v^*$ ▷ Horizontal propagation
  end while
  Output $\mathbf{Q}^* = \mathbf{Q}_h^*$ as the final converged pairwise constraint matrix

First, for the vertical constraint propagation, we suppose $\mathbf{Q}_v(0) = \mathbf{Z}$. Using the horizontal iteration equation, we can obtain,

$$\mathbf{Q}_v(t) = (\alpha\bar{\mathbf{A}})^{t-1}\mathbf{Z} + (1-\alpha)\sum_{i=0}^{t-1}(\alpha\bar{\mathbf{A}})^i\mathbf{Z} \tag{4}$$

Since the propagation parameter $0 < \alpha < 1$ and the eigenvalues of $\bar{\mathbf{A}}$ are in $[-1, 1]$, the horizontal propagation has a converged solution as below,

$$\lim_{t\to\infty}(\alpha\bar{\mathbf{A}})^{t-1} = 0 \tag{5}$$

$$\lim_{t\to\infty}\sum_{i=0}^{t-1}(\alpha\bar{\mathbf{A}})^i = (I-\alpha\bar{\mathbf{A}})^{-1} \tag{6}$$

$$\mathbf{Q}_v^* = \lim_{t\to\infty}\mathbf{Q}_v(t) = (1-\alpha)(I-\alpha\bar{\mathbf{A}})^{-1}\mathbf{Z} \tag{7}$$

Second, the horizontal constraint propagation can be transformed into a vertical propagation problem by a transpose operation,

$$\mathbf{Q}_h^\mathsf{T}(t+1) = \alpha\bar{\mathbf{A}}\mathbf{Q}_h^\mathsf{T}(t) + (1-\alpha)\mathbf{Q}_v^{*\mathsf{T}} \tag{8}$$

As shown in Eq. (7), the horizontal propagation converges to

$$\mathbf{Q}_h^{*\mathsf{T}} = (1-\alpha)(I-\alpha\bar{\mathbf{A}})^{-1}\mathbf{Q}_v^{*\mathsf{T}} \tag{9}$$

Therefore, the E2CP constraint propagation algorithm has the following closed-form feasible solution,

$$\begin{aligned}\mathbf{Q}^* = \mathbf{Q}_h^* &= (1-\alpha)\mathbf{Q}_v^*(I-\alpha\bar{\mathbf{A}}^\mathsf{T})^{-1} \\ &= (1-\alpha)^2(I-\alpha\bar{\mathbf{A}})^{-1}\mathbf{Z}(I-\alpha\bar{\mathbf{A}})^{-1}\end{aligned} \tag{10}$$

D. DETAILED FLOP/S ANALYSIS

We provided the total GFLOP/s of different diarization systems in Table 2, and discussed the results in Section 3.3. A detailed version of the FLOP/s analysis is provided in Table 3 and Table 4, where we break the FLOP/s numbers into different components.

For the speaker turn detection model and the speaker encoder model, the total number of FLOPs is estimated with TensorFlow Profiler by counting `total_float_ops`. For other components, the total number of FLOPs is estimated with PyPAPI by counting `PAPI_FP_OPS`. The denominator of FLOP/s is based on the length of audio being processed.

E. OPEN SOURCE PYTHON IMPLEMENTATION

We provide a Python-based open source library at https://github.com/wq2012/SpectralCluster, which covers these implementations:

1. Refinement operations on affinity matrix [7].
2. Laplacian matrix [27].
3. K-Means with cosine distance.
4. Auto-tune [28].
5. Constained spectral clustering with E2CP [30].

This library can be installed via `pip`:

```
pip3 install spectralcluster
```

The "Turn + E2CP + Auto-tune" configuration that produced the best performance in Table 2 is provided in `configs.py`. It can be directly used in the example below:

```
from spectralcluster import configs

labels = configs.turntodiarize_clusterer.predict(
    embeddings, constraint_matrix)
```

Table 3. GFLOPS/s of each component for different speaker diarization systems after running for 10min.

System	Method	Speaker turn detection	Speaker encoder	Eigen decomposition	E2CP	Laplacian & K-Means	Total
Dense d-vector	Dense	0	0.42	0.43	0	0.00	0.85
	Dense + Auto-tune	0	0.42	4.34	0	0.00	4.76
Turn-to-diarize	Turn	0.58	0.42	0.00	0	0.00	1.00
	Turn + E2CP	0.58	0.42	0.00	0.00	0.00	1.00
	Turn + Auto-tune	0.58	0.42	0.03	0	0.00	1.02
	Turn + E2CP + Auto-tune	0.58	0.42	0.03	0.00	0.00	1.02

Table 4. GFLOPS/s of each component for different speaker diarization systems after running for 1h.

System	Method	Speaker turn detection	Speaker encoder	Eigen decomposition	E2CP	Laplacian & K-Means	Total
Dense d-vector	Dense	0	0.42	36.09	0	0.04	36.54
	Dense + Auto-tune	0	0.42	360.92	0	0.04	361.37
Turn-to-diarize	Turn	0.58	0.42	0.18	0	0.00	1.18
	Turn + E2CP	0.58	0.42	0.18	0.00	0.00	1.18
	Turn + Auto-tune	0.58	0.42	1.82	0	0.00	2.81
	Turn + E2CP + Auto-tune	0.58	0.42	1.82	0.00	0.00	2.81

However, we also want to clarify that our optimal clustering configuration is based on our specific speaker turn detection and speaker encoder models. The clustering configuration may need to be adjusted when using different models.