Interspeech 2024
1-5 September 2024, Kos, Greece

DiarizationLM: Speaker Diarization Post-Processing with Large Language Models

Quan Wang, Yiling Huang*, Guanlong Zhao*, Evan Clark, Wei Xia, Hank Liao*

Google LLC, USA

{quanw, yilinghuang, guanlongzhao, evclark, ericwxia, hankliao}@google.com

Abstract

In this paper, we introduce DiarizationLM, a framework to leverage large language models (LLM) to post-process the outputs from a speaker diarization system. In this framework, the outputs of the automatic speech recognition (ASR) and speaker diarization systems are represented as a compact textual format, which is included in the prompt to an optionally finetuned LLM. The outputs of the LLM can be used as the refined diarization results with the desired enhancement. As a post-processing step, this framework can be easily applied to any off-the-shelf ASR and speaker diarization systems without retraining existing components. Our experiments show that a finetuned PaLM 2-S model can reduce the WDER by rel. 55.5% on the Fisher telephone conversation dataset, and rel. 44.9% on the Callhome English dataset.

Index Terms: speaker diarization, large language models

1. Introduction

Speaker diarization is the task of partitioning speech into homogeneous segments according to speaker identities, answering the question "who spoken when" [1]. Typical speaker diarization systems can be roughly categorized into two groups: modularized systems and end-to-end systems. A modularized speaker diarization system usually consists of multiple separately trained components including voice activity detection (VAD) [2], speaker turn detection [3, 4], speaker encoder [5], and a clustering algorithm, which can be either unsupervised [6–9] or supervised [10, 11]. End-to-end systems, on the other hand, directly optimize the entire system on diarization errors by introducing a permutation invariant loss function [12, 13].

In many real world applications such as meeting summarization, call center analysis, mobile recorder apps [14], and video captioning, knowing "who spoke when" is not sufficient. Speaker labels are more interpretable and meaningful when they are associated with speech transcripts. Various solutions have been proposed to directly address the problem of "who spoke what," including jointly training speech recognition and speaker diarization [15], speaker-attributed automatic speech recognition (SA-ASR) [16], target speaker automatic speech recognition (TS-ASR) [17–19] and word-level end-to-end neural speaker diarization [20].

In practice, however, most production speech systems still consist of separately trained ASR models and speaker diarization models, with various considerations including:

1. *Modularized development and deployment:* ASR and speaker diarization systems are usually trained on different datasets, and potentially using different modeling framework, by different research teams.
2. *Potential quality regression on ASR:* ASR has many more use cases than speaker diarization. Joint modeling of ASR and speaker diarization usually has worse Word Error Rates (WER) than ASR-only models, thus is not acceptable in many applications.
3. *Flexibility:* Combining separately trained ASR models and speaker diarization models is a very flexible solution. As long as the ASR model provides word timing information, it can be combined with almost any speaker diarization model, either unsupervised or supervised, either modularized or end-to-end trained.

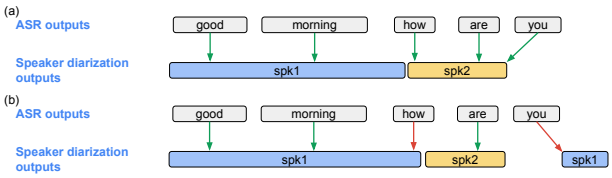

Figure 1: *The orchestration module associates each word from the ASR transcript with a speaker label from the speaker diarization outputs. (a) In this example, all words are associated with the correct speaker labels (green arrows). (b) In this example, two words are associated with wrong speaker labels (red arrows) due to inconsistent timing information from the two systems.*

We refer to the combination of ASR transcripts and speaker diarization results as an *orchestration module* (in some other work [21], this process is called "reconciliation"). In this module, each word from the ASR transcript is associated with a speaker label. A typical orchestration algorithm works as follows: (1) If the word segment overlaps with at least one speaker segment, then this word is associated with the speaker that has the biggest temporal overlap with this word; (2) otherwise if this word segment does not overlap with any speaker segment, then it is associated with the speaker that has the smallest temporal distance to this word based on the segment boundaries. This orchestration algorithm is illustrated in Fig. 1 (a).

However, since ASR and speaker diarization are separately trained with usually different training datasets and modeling approaches, the timing information from these two systems can be inconsistent, resulting in word diarization errors, as demonstrated with the example in Fig. 1 (b). Specifically, modern ASR models are usually trained end-to-end without using the ground truth timing information, and the word timing is inferred from the probability lattice of the decoder, which could be inaccurate.

In many cases, such errors can usually be fixed by leveraging semantic information from the ASR transcripts. Take Fig. 1 as an example, simply by looking at the textual transcript "good

*Equal contribution.

10.21437/Interspeech.2024-209

morning how are you," if we know it consists of two speakers, we can easily tell which word comes from which speaker confidently without using any *acoustic* speaker diarization system. In practice, diarization errors can be much more complicated than the simple example in Fig. 1. To handle such cases, we propose DiarizationLM, a framework to post-process the orchestrated ASR and speaker diarization outputs with a large language model (LLM). In this work, we only focus on LLM's capability to reduce diarization errors.

2. Related work

Speaker diarization post-processing: In the context of conventional speaker diarization, "post-processing" usually refers to a stage where the clustering results are refined with signals from other sources or systems. An early post-processing approach was known as "resegmentation", where the Gaussian mixture models (GMMs) are estimated for each speaker with the Baum-Welch algorithm, and a Viterbi algorithm is used to re-annotate the speakers with the GMMs [22]. Later in [23], the authors proposed to use a neural network for resegmentation, with an additional class for non-speech. In [24], the authors proposed DiaCorrect, a method inspired by error correction techniques in ASR. One major difference in our proposed framework is that we leverage semantic information to refine the diarization results on a word level, while these resegmentation approaches are only based on acoustic information and perform at the cluster level.

Speaker diarization with LLM: In [21], word embeddings from the ASR transcript are extracted with a pre-trained Roberta-base LM [25]. Then a separately trained transformer encoder takes the word embeddings and predicted speaker labels as input and produces the corrected speaker labels. The transformer encoder is trained on both simulated diarization errors and real data. The biggest difference between the proposed framework with [21] is that we directly feed the compact pure textual representation of the ASR and diarization results as part of the prompt to the LLM, and directly finetune the LLM to produce the corrected results in the same compact textual representation, hence the DiarizationLM does not rely on internal embedding representations from the LLM. More recently in [26], the authors use an LLM to predict the speaker probability for the next word, and incorporate this probability into the beam search decoding of speaker diarization. The proposed framework differs from [26] because it can be more generally applied to any speaker diarization system, instead of requiring an internal state (word-level speaker probabilities) for the beam search decoding process.

3. DiarizationLM

3.1. System overview

We illustrate the DiarizationLM in Fig. 2. In DiarizationLM, the ASR and speaker diarization systems are frozen. Their outputs are processed by the orchestration module to associate a speaker label with each recognized word. The orchestrated diarization outputs are processed by a *prompt builder* module, which creates a compact textual representation of the diarized transcript, segments the input into shorter chunks to fit the LLM input size limit, and applies the prompt prefix and suffix. The prompts are then sent to a finetuned LLM, and the completions generated by the LLM are handled by a *completion parser* module, which truncates undesired outputs from the LLM, combines the completions of multiple segments (i.e., text chunks), and applies a transform (see Section 3.4) to preserve the original transcripts of the ASR model.

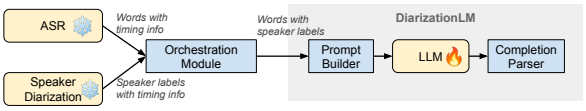

Figure 2: *Diagram of the proposed DiarizationLM framework.*

3.2. Prompt builder

The output of the orchestration module is two sequences of equal length: a sequence of words, and a sequence of speaker labels. To fit them into a prompt, we use a compact representation, where speaker tokens are inserted in the beginning of the transcript or speaker turns, e.g., "<spk:1> howdy <spk:2> how are you."

Since most LLMs have an input length limit, the text representation of an entire longform conversation may not fit this limit. In such cases, we recursively binary partition the word and speaker sequences in the middle, until all segments fit the the input length limit. We also apply prefix and suffix to each prompt. The prefix is usually an instruction describing the task for the LLM to perform, and the suffix is a sequence of tokens to indicate the end of the prompt.

3.3. Completion parser

Each prompt from the prompt builder is sent to the finetuned LLM, which generates a text completion for this prompt. First of all, we truncate any undesired outputs after the suffix that represents the end of completion. Then, we convert the text representation of the completion back to the word sequence and the speaker sequence format. If the text representation does not start with a speaker token, we either use the last speaker from the previous segment, or use speaker 1 if it is the first segment. Next, we concatenate the word sequences and speaker sequences from all segments. However, the resulting concatenated word sequence may not be identical to the original word sequence from the ASR model due to modifications by the LLM. This is undesired and may hurt word error rate. Thus here we need an algorithm to transfer the LLM-corrected speaker labels to the original word sequence from the ASR model. We introduce this algorithm in the next section.

3.4. Transcript-Preserving Speaker Transfer

Here we describe an algorithm called *Transcript-Preserving Speaker Transfer* (TPST)[1], which is used in several places in the proposed DiarizationLM framework, including training data preparation and the completion parser module.

Assuming that there are two diarized transcripts, referred to as "source" and "target", each represented by two sequences of the same length: a sequence of words, and a sequence of speaker labels. The purpose of TPST is to transfer the speaker labels from the source sequence to the target sequence, such that (1) the transferred speaker label sequence has a 1-to-1 association with the target word sequence; (2) the transferred speaker labels are more consistent with the source speaker labels. For example, the concatenated word sequence from the completion parser module may not be identical to the original word sequence from the ASR model. Thus we can treat the completion sequences as the source, and the original sequences from the orchestration module as the target, and transfer the speaker labels.

We compile the DiarizationLM output as the original ASR words associated with the TPST-transferred speaker labels. The

[1] https://github.com/google/speaker-id/tree/master/DiarizationLM

detailed TPST algorithm is described in Algorithm 1.

Algorithm 1 The transcript-preserving speaker transfer (TPST) algorithm.

inputs
Source word sequence of length N: $\mathbf{w}^{src} = (w_1^{src}, \cdots, w_N^{src})$
Source speaker sequence of length N: $\mathbf{s}^{src} = (s_1^{src}, \cdots, s_N^{src})$
Target word sequence of length M: $\mathbf{w}^{tgt} = (w_1^{tgt}, \cdots, w_M^{tgt})$
Target speaker sequence of length M: $\mathbf{s}^{tgt} = (s_1^{tgt}, \cdots, s_M^{tgt})$

outputs
Transfered speaker sequence of length M: $\mathbf{s}^{tra} = (s_1^{tra}, \cdots, s_M^{tra})$

procedure TPST($\mathbf{w}^{src}, \mathbf{s}^{src}, \mathbf{w}^{tgt}, \mathbf{s}^{tgt}$)
  Align $\mathbf{w}^{src}$ to $\mathbf{w}^{tgt}$ with the Levenshtein algorithm [27]
  $\mathbf{s}^{ali} \leftarrow f_{align}(\mathbf{s}^{src})$ ▷ $\mathbf{s}^{ali}$ is a speaker sequence of length M
  $K \leftarrow \max\{\max(\mathbf{s}^{ali}), \max(\mathbf{s}^{tgt})\}$ ▷ max number of speakers
  Initialize a cost matrix $\mathbf{C} \in \mathbb{R}^{K \times K}$
  for $1 \leq i \leq K$ and $1 \leq j \leq K$ **do**
    $\mathbf{C}_{i,j} \leftarrow \sum_{1 \leq m \leq M} \delta(s_m^{ali} = i;$ and; $s_m^{tgt} = j)$
  end for
  Solve assign. problem with cost matrix $\mathbf{C}$ using Hungarian algorithm [28]
  Resulting in a transform $f_{assign}(\cdot)$ ▷ handle speaker permutations
  for $1 \leq m \leq M$ **do**
    if $s_m^{ali} \neq \emptyset$ **then**
      $s_m^{tra} \leftarrow f_{assign}(s_m^{ali})$ ▷ transfer the speakers from the src
    else
      $s_m^{tra} \leftarrow s_m^{tgt}$ ▷ preserve the tgt speaker if no src speaker
    end if
  end for
end procedure

3.5. LLM finetuning

To finetune the LLM, we build the training data as a collection of prompt-completion pairs. First, for each utterance, we run the ASR model and the speaker diarization system on it, and apply the orchestration module as shown in Fig. 2. This produces the hypothesis word sequence $\mathbf{w}^{hyp}$ and hypothesis speaker sequence $\mathbf{s}^{hyp}$. From the ground truth annotations of this utterance, we build the reference word sequence $\mathbf{w}^{ref}$ and the reference speaker sequence $\mathbf{s}^{ref}$. Given these four sequences, we can build the prompts and completions in the training data with three different flavors, as introduced below.

Note that we cannot train a model to directly map the hypothesis to the raw ground-truth, which requires correcting ASR and diarization errors at the same time and is out of the scope of this work. For all three flavors, it is critical for the prompt and completion to use the same word sequence with different speaker sequences. This helps the LLM to focus on correcting the speaker labels without modifying the ASR transcripts.

Flavor 1: *hypothesis-to-oracle*, or simply *hyp2ora*. In this flavor, we apply the TPST algorithm by treating reference sequences as source and hypothesis sequences as target:

$$\mathbf{s}^{ora} = \mathrm{TPST}(\mathbf{w}^{ref}, \mathbf{s}^{ref}, \mathbf{w}^{hyp}, \mathbf{s}^{hyp}), \tag{1}$$

where the output $\mathbf{s}^{ora}$ is the **oracle hypothesis speakers** transfered from the reference sequences. With $\mathbf{s}^{ora}$, the prompts and completions in our training data are created as below:

- *Prompts*: The text representation of $\mathbf{w}^{hyp}$ and $\mathbf{s}^{hyp}$, with segmentation, and optionally prefix and suffix.
- *Completions*: The text representation of $\mathbf{w}^{hyp}$ and $\mathbf{s}^{ora}$, with segmentation, and optionally suffix.

Flavor 2: *degraded-to-reference*, or simply *deg2ref*. In this flavor, we apply the TPST algorithm by treating hypothesis sequences as source and reference sequences as target:

$$\mathbf{s}^{deg} = \mathrm{TPST}(\mathbf{w}^{hyp}, \mathbf{s}^{hyp}, \mathbf{w}^{ref}, \mathbf{s}^{ref}), \tag{2}$$

where the output $\mathbf{s}^{deg}$ is the **degraded reference speakers** transfered from the hypothesis sequences. With $\mathbf{s}^{deg}$, the prompts and completions in our training data are created as below:

- *Prompts*: The text representation of $\mathbf{w}^{ref}$ and $\mathbf{s}^{deg}$, with segmentation, and optionally prefix and suffix.
- *Completions*: The text representation of $\mathbf{w}^{ref}$ and $\mathbf{s}^{ref}$, with segmentation, and optionally suffix.

Flavor 3: *mixed*. This is simply the union of the prompts and completions from the previous two flavors. When building training batches, prompt-completion pairs from the two flavors are interleaved.

4. Experiments

4.1. Datasets

To finetune the LLM, we use the training subset of the Fisher corpus [29], which consists of 1,920 hours of 11,527 conversations. The same train-test split of the Fisher dataset has been used in prior works [4, 9, 21, 30]. For evaluation, we use the testing subset of the Fisher corpus [29], as well as the testing subset of Callhome American English data [31]. The Fisher testing subset consists of 28.7 hours of 172 conversations[2]. The Callhome American English testing subset consists of 1.7 hours of 20 conversations. Both datasets are in the telephony speech domain. All conversations have 2 speakers.

4.2. Metrics

To evaluate the diarization performance, we use two metrics: the Word Diarization Error Rate (WDER) [15] and the concatenated minimum-permutation Word Error Rate (cpWER) [32]. To briefly recap, WDER is defined as:

$$\mathrm{WDER} = \frac{S_{\mathrm{IS}} + C_{\mathrm{IS}}}{S + C}, \tag{3}$$

where S_{IS} is the number of ASR substitutions with incorrect speaker labels, C_{IS} is the number of correct ASR words with incorrect speaker labels, S is the number of ASR substitutions, and C is the number of correct ASR words.

cpWER is computed as follows: (1) Concatenate all transcripts of each speaker for both reference and hypothesis. (2) Compute the WER between the reference and all possible speaker permutations of the hypothesis. (3) Pick the lowest WER among all these permutations, which is considered as the best permutation.

4.3. Models

For the ASR model in Fig. 2, we use a Universal Speech Model (USM) [33] with 600 million parameters trained with the RNN-T loss [34]. For the speaker diarization model in Fig. 2, we use the turn-to-diarize system [3] with a multi-stage clustering setup [9] in the experiments. The number of speakers is unknown (from 1 to ∞) to the speaker diarization system in all of the experiments. However, we would like to point out that the proposed framework is generic and should work with other ASR or speaker diarization systems as well, such as variants of end-to-end speaker diarization models [12, 13].

For the LLM in Fig. 2, we experiment with the PaLM 2-S model ("`text-bison`" model in Google Cloud API) and the PaLM 2-L model ("`text-unicorn`" model in Google Cloud

[2] `https://github.com/google/speaker-id/blob/master/publications/ScdLoss/eval/fisher.txt`

Table 1: *Evaluation results of the USM + turn-to-diarize baseline system and the results post-processed by DiarizationLM. WERs are the same for all systems due to TPST.*

System	Fisher testing set			Callhome testing set		
	WER	WDER	cpWER	WER	WDER	cpWER
USM + turn-to-diarize baseline	15.48	5.32	21.19	15.36	7.72	24.39
+ PaLM 2-S zero-shot	-	11.96	30.19	-	12.26	30.60
+ PaLM 2-S one-shot	-	16.58	38.03	-	14.50	34.32
+ PaLM 2-L zero-shot	-	11.36	31.78	-	13.29	34.30
+ PaLM 2-L one-shot	-	5.94	22.21	-	7.95	24.67
+ PaLM 2-S finetuned (hyp2ora flavor)	-	**2.37**	**16.93**	-	**4.25**	**20.22**
+ PaLM 2-S finetuned (deg2ref flavor)	-	3.94	18.55	-	5.33	21.47
+ PaLM 2-S finetuned (mixed flavor)	-	2.41	16.94	-	4.76	20.84

Table 2: *Evaluation results of the turn-to-diarize baseline system with reference ASR transcript (assuming WER=0%) and the results post-processed by DiarizationLM.*

System	Fisher testing set		Callhome testing set	
	WDER	cpWER	WDER	cpWER
Reference + turn-to-diarize baseline	2.81	5.19	3.74	6.82
+ PaLM 2-S zero-shot	7.50	12.70	7.29	12.79
+ PaLM 2-S one-shot	10.92	19.16	12.79	21.65
+ PaLM 2-L zero-shot	8.69	16.85	11.67	22.87
+ PaLM 2-L one-shot	3.23	5.99	3.76	6.95
+ PaLM 2-S finetuned	**1.18**	**2.21**	**1.49**	**2.66**

API) [35]. We use the PaLM 2-S model as the foundation model, and finetune it on the dataset described in Section 4.1 with data processing steps described in Section 3.5. This model uses a sentence piece model (SPM) of 256k tokens as its tokenizer [36]. During finetuning, we limit the LLM input size to 4,096 tokens, and segment the training and testing data accordingly. The PaLM 2-L model is only used for zero-shot and one-shot experiments, as described in Section 4.4.

In the prompt builder module, we use an empty prompt prefix, and a 5-character prompt suffix "␣--->␣" (note the two spaces surrounding the arrow). For the completions in the training data, we use a 6-character completion suffix "␣[eod]" (short for "end of document"; note the leading space). After processing the training data with the prompt builder module, we result in 13,430 prompt-completion pairs for training in total. The average length of a prompt is 2,371 SPM tokens, and the average length of a completion is 2,329 tokens. The LLM is finetuned for 1,200 steps with a batch size of 16.

4.4. Zero-shot and one-shot baselines

Apart from finetuning the PaLM 2-S model on the speaker diarization task, we also experiment with directly using the PaLM 2-S and PaLM 2-L models on the speaker diarization task without finetuning (i.e., zero-shot or few-shot).

For the zero-shot setup, we use the prompt "*In the speaker diarization transcript below, some words are potentially misplaced. Please correct those words and move them to the right speaker. Directly show the corrected transcript without explaining what changes were made or why you made those changes.*"

For the one-shot setup, the prompt contains task instructions and an example, i.e., "*In the speaker diarization transcript below, some words are potentially misplaced. Please correct those words and move them to the right speaker. For example, given this input transcript, '<spk:1> How are you doing today? I <spk:2> am doing very well. How was everything at the <spk:1> party? Oh, the party? It was awesome. We had lots of fun. Good <spk:2> to hear!' The correct output transcript should be: '<spk:1> How are you doing today? <spk:2> I am doing very well. How was everything at the party? <spk:1> Oh, the party? It was awesome. We had lots of fun. <spk:2> Good to hear!' Now, please correct the transcript below.*"

4.5. Evaluation results

In Table 1, we show the evaluation results of the USM + turn-to-diarize baseline together with the outputs post-processed by DiarizationLM. We report results for zero-shot, one-shot, and finetuning on the diarization task with three different flavors.

For zero-shot and one-shot experiments with PaLM 2-S, we observe significantly worse WDER and cpWER performance compared with the baseline system, indicating that the PaLM 2-S foundation model does not offer speaker diarization capabilities without finetuning. Zero-shot experiment with PaLM 2-L model does not show performance gains either. Although PaLM 2-L one-shot results show improvements over the zero-shot configuration, the results are still worse than the baseline system. Manually inspecting the per-utterance results suggests that the PaLM 2-L model with one-shot is able to improve speaker diarization in relatively simple cases. However, real world applications can be more complicated with errors from both the ASR system and the speaker diarization system. In such cases, even with one-shot, LLM can still introduce even more errors to the results if not finetuned specifically on the speaker diarization task.

On both datasets, we observe big improvements of both WDER and cpWER with any of the three finetuning flavors. Interestingly, the biggest improvement is observed with the *hyp2ora* flavor, while the smallest improvement is observed with the *deg2ref* flavor. Specifically, for *hyp2ora*, we see a rel. 55.5% improvement of WDER after post-processing with DiarizationLM on the Fisher testing set. Although we do not use any Callhome data during the LLM finetuning, we see a rel. 44.9% improvement of WDER on the Callhome testing set. The WER of the USM on the two testing sets are relatively high due to domain mismatch and suboptimal annotation quality of the ground truth. However, this also demonstrates that the DiarizationLM solution provides consistent quality gains even with out-of-domain ASR and speaker diarization models.

To analyze DiarizationLM's performance without the interference of ASR errors, in Table 2, we show the results of a similar setup but replacing the ASR transcripts with the ground truth transcripts. For these experiments, by definition, we have WER=0%, and the *hyp2ora* and *deg2ref* flavors are equivalent. We again see big improvements of WDER after post-processing the diarization results by the same DiarizationLM model (i.e. the *deg2ref* flavor in Table 1).

5. Discussion and Conclusion

The experiments in Section 4 show promising results where LLMs can significantly reduce speaker diarization errors. However, we admit the limitations of these experiments. The training and testing data for the experiments are all based on the telephony speech domain, all with exactly 2 speakers. An important future work would be to include more diverse datasets to finetune the LLM and evaluate its performance across different domains with varied number of speakers. Another future research direction is to compare LLMs with different size variants on the speaker diarization task. Specifically, the performance will likely be even better if we finetune larger models such as PaLM 2-M or PaLM 2-L. It would also be interesting to reproduce the experiments with other speaker diarization systems such as EEND [12] or WEEND [20]. Lastly, as PaLM 2 models are multilingual [35], the DiarizationLM framework can be naturally applied to speaker diarization tasks in many languages. It would be helpful to evaluate how DiarizationLM performs on speaker diarization datasets in languages other than English.

6. References

[1] 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*, vol. 72, p. 101317, 2022.

[2] S. Ding, R. Rikhye, Q. Liang, Y. He, Q. Wang, A. Narayanan, T. O'Malley, and I. McGraw, "Personal VAD 2.0: Optimizing Personal Voice Activity Detection for On-Device Speech Recognition," in *Proc. Interspeech*, 2022, pp. 3744–3748.

[3] W. Xia, H. Lu, Q. Wang, A. Tripathi, Y. Huang, I. L. Moreno, and H. Sak, "Turn-to-Diarize: Online speaker diarization constrained by transformer transducer speaker turn detection," in *Proc. ICASSP*, 2022, pp. 8077–8081.

[4] G. Zhao, Q. Wang, H. Lu, Y. Huang, and I. L. Moreno, "Augmenting transformer-transducer based speaker change detection with token-level training loss," in *Proc. ICASSP*, 2023.

[5] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. ICASSP*, 2018, pp. 4879–4883.

[6] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. Lopez Moreno, "Speaker diarization with LSTM," in *Proc. ICASSP*, 2018, pp. 5239–5243.

[7] T. J. Park, K. J. Han, M. Kumar, and S. Narayanan, "Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap," *IEEE Signal Processing Letters*, vol. 27, pp. 381–385, 2019.

[8] T. J. Park, M. Kumar, and S. Narayanan, "Multi-scale speaker diarization with neural affinity score fusion," in *Proc. ICASSP*, 2021, pp. 7173–7177.

[9] Q. Wang, Y. Huang, H. Lu, G. Zhao, and I. L. Moreno, "Highly efficient real-time streaming and fully on-device speaker diarization with multi-stage clustering," *arXiv preprint arXiv:2210.13690*, 2022.

[10] A. Zhang, Q. Wang, Z. Zhu, J. Paisley, and C. Wang, "Fully supervised speaker diarization," in *Proc. ICASSP*, 2019, pp. 6301–6305.

[11] Q. Li, F. L. Kreyssig, C. Zhang, and P. C. Woodland, "Discriminative neural clustering for speaker diarisation," in *Proc. Spoken Language Technology Workshop*, 2021, pp. 574–581.

[12] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe, "End-to-end neural speaker diarization with permutation-free objectives," in *Proc. Interspeech*, 2019, pp. 4300–4304.

[13] S. Maiti, Y. Ueda, S. Watanabe, C. Zhang, M. Yu, S.-X. Zhang, and Y. Xu, "Eend-ss: Joint end-to-end neural speaker diarization and speech separation for flexible number of speakers," in *Proc. Spoken Language Technology Workshop*, 2023, pp. 480–487.

[14] Q. Wang and F. Zhang, "Who said what? Recorder's on-device solution for labeling speakers," Google AI Blog.

[15] L. E. Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," in *Proc. Interspeech*, 2019, pp. 396–400.

[16] N. Kanda, J. Wu, Y. Wu, X. Xiao, Z. Meng, X. Wang, Y. Gaur, Z. Chen, J. Li, and T. Yoshioka, "Streaming Speaker-Attributed ASR with Token-Level Speaker Embeddings," in *Proc. Interspeech*, 2022, pp. 521–525.

[17] K. Zmolikova, M. Delcroix, K. Kinoshita, T. Higuchi, A. Ogawa, and T. Nakatani, "Speaker-aware neural network based beamformer for speaker extraction in speech mixtures," in *Proc. Interspeech*, 2017, pp. 2655–2659.

[18] M. Delcroix, S. Watanabe, T. Ochiai, K. Kinoshita, S. Karita, A. Ogawa, and T. Nakatani, "End-to-end SpeakerBeam for single channel target speech recognition," in *Proc. Interspeech*, 2019, pp. 451–455.

[19] N. Kanda, S. Horiguchi, R. Takashima, Y. Fujita, K. Nagamatsu, and S. Watanabe, "Auxiliary Interference Speaker Loss for Target-Speaker Speech Recognition," in *Proc. Interspeech*, 2019, pp. 236–240.

[20] Y. Huang, W. Wang, G. Zhao, H. Liao, W. Xia, and Q. Wang, "Towards word-level end-to-end neural speaker diarization with auxiliary network," *arXiv preprint arXiv:2309.08489*, 2023.

[21] R. Paturi, S. Srinivasan, and X. Li, "Lexical Speaker Error Correction: Leveraging Language Models for Speaker Diarization Error Correction," in *Proc. Interspeech*, 2023, pp. 3567–3571.

[22] G. Sell and D. Garcia-Romero, "Diarization resegmentation in the factor analysis subspace," in *Proc. ICASSP*, 2015, pp. 4794–4798.

[23] R. Yin, H. Bredin, and C. Barras, "Neural speech turn segmentation and affinity propagation for speaker diarization," in *Proc. Interspeech*, 2018, pp. 1393–1397.

[24] J. Han, F. Landini, J. Rohdin, M. Diez, L. Burget, Y. Cao, H. Lu, and J. Cernocky, "DiaCorrect: Error correction back-end for speaker diarization," *arXiv preprint arXiv:2309.08377*, 2023.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[26] T. J. Park, K. Dhawan, N. Koluguri, and J. Balam, "Enhancing speaker diarization with large language models: A contextual beam search approach," *arXiv preprint arXiv:2309.05248*, 2023.

[27] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[28] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

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

[30] G. Zhao, Y. Wang, J. Pelecanos, Y. Zhang, H. Liao, Y. Huang, H. Lu, and Q. Wang, "USM-SCD: Multilingual speaker change detection based on large pretrained foundation models," *arXiv preprint arXiv:2309.08023*, 2023.

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

[32] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, D. Raj *et al.*, "CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings," *arXiv preprint arXiv:2004.09249*, 2020.

[33] Y. Zhang, W. Han, J. Qin, Y. Wang, A. Bapna, Z. Chen, N. Chen, B. Li, V. Axelrod, G. Wang *et al.*, "Google USM: Scaling automatic speech recognition beyond 100 languages," *arXiv preprint arXiv:2303.01037*, 2023.

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

[35] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "PaLM 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[36] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.