Google

DiarizationLM: Speaker Diarization Post-Processing with Large Language Models

Quan Wang[*]   **Yiling Huang**[*]   **Guanlong Zhao**[*]
Evan Clark   **Wei Xia**   **Hank Liao**
Google LLC, USA   [*]Equal contribution
✉ quanw@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. Various goals can be achieved with the proposed framework, such as improving the readability of the diarized transcript, or reducing the word diarization error rate (WDER). 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.

Code: https://github.com/google/speaker-id/tree/master/DiarizationLM

Model: https://huggingface.co/google/DiarizationLM-8b-Fisher-v2

Demo: https://huggingface.co/spaces/diarizers-community/DiarizationLM-GGUF

Contents

1 INTRODUCTION

Speaker diarization is the task of partitioning speech into homogeneous segments according to speaker identities, answering the question "who spoken when" [1, 2]. 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) [3, 4, 5, 6], speaker turn detection [7, 8], speaker encoder [9, 10, 11], and a clustering algorithm, which can be either unsupervised [12, 13, 14, 15, 16, 17] or supervised [18, 19]. End-to-end systems , on the other hand, directly optimize the entire system on diarization errors by introducing a permutation invariant loss function [20, 21, 22, 23].

In many real world applications such as meeting summarization, call center analysis, mobile recorder apps [24], 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 [25], speaker-attributed automatic speech recognition (SA-ASR) [26, 27, 28, 29], target speaker automatic speech recognition (TS-ASR) [30, 31, 32, 33] and word-level end-to-end neural speaker diarization [34].

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.
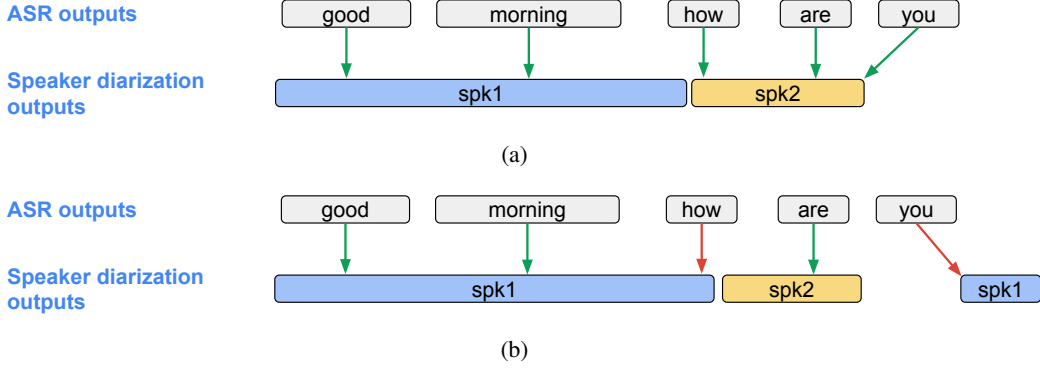
Figure 1: The orchestration module associates each word from the ASR transcript with a speaker label from the speaker diarization outputs. (a) In this example, all words are associated with the correct speaker labels (green arrows). The words "good", "morning", and "are" and "you" are associated with the only speaker label that overlap with them. The word "how" overlaps with both spk1 and spk2, but has bigger overlaps with spk2, thus is associated with spk2. The word "you" does not overlap with any speaker, but is closest to spk2, thus is associated with spk2. (b) In this example, two words are associated with wrong speaker labels (red arrows) due to inconsistent timing information from the two systems. The word "how" is mistakenly associated with spk1, since spk1 has more overlap with this word than spk2. The word "you" is mistakenly associated with spk1, since spk1 is closer to this word than spk2.

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.

We refer to the combination of ASR transcripts and speaker diarization results as an *orchestration module* (in some other work [35], 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. 1a.

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. 1b. 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 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). While the experiments performed in this paper mainly focus on reducing word diarization errors using LLM, we also show examples how LLM can be used for other purposes such as autofilling speaker names, autofilling speaker roles, or even completely replacing the orchestration module in Appendix B.

2   MOTIVATING EXAMPLE

Here is a small example where we construct a prompt (in blue) which consists of a brief instruction, and the diarization transcript with several errors. The prompt is sent to Google Bard [36] via its Web UI[1]. We can see that the completion generated by LLM (in orange) correctly moved the error words to the right speakers. This example can be easily reproduced with many off-the-shelf commercial or open source instruction-finetuned LLMs, such as Google Bard [36] or OpenAI ChatGPT [37].

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.

<spk:1> Good morning Patrick, how
<spk:2> are you? Good, good. How are you Tom? Pretty
<spk:1> good. Going to work?
<spk:2> Yes. Busy day. How are your kids? Do they go
<spk:1> to school? Oh they are too young for that. I sent them to daycare earlier
<spk:2> today. Oh yeah I forgot about that.

Completion:

<spk:1> Good morning Patrick, how are you?
<spk:2> Good, good. How are you Tom?
<spk:1> Pretty good. Going to work?
<spk:2> Yes. Busy day. How are your kids? Do they go to school?
<spk:1> Oh, they are too young for that. I sent them to daycare earlier today.
<spk:2> Oh yeah, I forgot about that.

Apart from correcting diarization errors, we also found several other diarization capabilities of LLMs, such as autofilling speaker names, autofilling speaker roles, or even completely replacing the orchestration module. Examples of these capabilities are shown in Appendix B. However, due to lack of relevant training and testing datasets, these capabilities are not thoroughly studied or evaluated in this paper. Here we focus on LLM's capability of reducing diarization errors.

3   DIARIZATIONLM

3.1   SYSTEM OVERVIEW

We illustrate the DiarizationLM framework in Fig. 2. In this framework, the ASR and speaker diarization systems are frozen, and 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, segment it into shorter versions to fit the LLM input size limit, and apply prompt prefix and suffix. The prompts are then sent to a finetuned LLM, and the completions generated by the LLM will be handled by a *completion parser* module, which truncates undesired outputs from the LLM, combines the completions of multiple segments, and apply a transform (see Section 3.4) to preserve the original transcripts of the ASR model.

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 it into a prompt, we use a compact textual representation, where speaker tokens are only inserted in the beginning of the transcript, or when the speaker has changed. Below is an example:

---

[1]We used an internal version of Bard that is based on a larger model and supports more tokens than the public version.
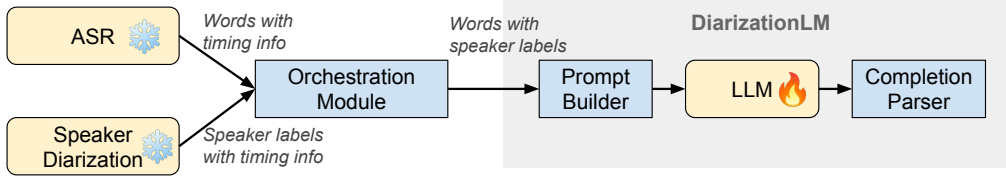
Figure 2: Diagram of the proposed DiarizationLM framework.

Example text representation of the prompt

```
Word sequence:          ["good", "morning", "how", "are", "you"]
Speaker sequence:       [1, 1, 2, 2, 2]
Text representation:    "<spk:1> good morning <spk:2> how are you"
```

Since most LLMs have an input length limit, the text representation of an entire utterance 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 will be sent to the finetuned LLM, which will generate a text completion for this prompt. First of all, we need to truncate any undesired outputs from the LLM. For example, during the LLM finetuning, each completion may have a suffix to indicate the end of the completion. Thus the suffix and any text generated after the suffix should be truncated from the original completion.

After the truncation, we need to 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 just 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 LLM. This is undesired and may hurt word error rate. Thus here we need an algorithm to transfer the speaker labels from the concatenated speaker sequence to the original word sequence from the ASR model. We will introduce this algorithm in the following section.

3.4 TRANSCRIPT-PRESERVING SPEAKER TRANSFER

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

Assume we have two sets of diarized transcript, 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 sequences to the target sequences, 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.

5

As an 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. Finally, the DiarizationLM outputs will be the original word sequence, associated with the transferred speaker label sequence.

The detailed TPST algorithm is described in Algorithm 1. An implementation is open sourced on GitHub [2].

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

1: **procedure** TPST($\mathbf{w}^{src}, \mathbf{s}^{src}, \mathbf{w}^{tgt}, \mathbf{s}^{tgt}$)
2:     Align $\mathbf{w}^{src}$ to $\mathbf{w}^{tgt}$ with the Levenshtein algorithm [38], resulting in a transform $f_{align}(\cdot)$
3:     $\mathbf{s}^{ali} \leftarrow f_{align}(\mathbf{s}^{src})$ ▷ $\mathbf{s}^{ali}$ is a speaker sequence of length M, and may contain blank speakers $\emptyset$ due to insertion errors in the alignment
4:     $K \leftarrow \max\{\max(\mathbf{s}^{ali}), \max(\mathbf{s}^{tgt})\}$ ▷ the maximal number of speakers in $\mathbf{s}^{ali}$ and $\mathbf{s}^{tgt}$
5:     Initialize a cost matrix $\mathbf{C} \in \mathbb{R}^{K \times K}$
6:     **for** $1 \leq i \leq K$ and $1 \leq j \leq K$ **do**
7:         $\mathbf{C}_{i,j} \leftarrow \sum_{1 \leq m \leq M} \delta(s_m^{ali} = i \text{ and } s_m^{tgt} = j)$
8:     **end for**
9:     Solve the assignment problem with cost matrix $\mathbf{C}$ using the Hungarian algorithm [39], resulting in a transform $f_{assign}(\cdot)$ ▷ handle speaker permutations
10:     **for** $1 \leq m \leq M$ **do**
11:         **if** $s_m^{ali} \neq \emptyset$ **then**
12:             $s_m^{tra} \leftarrow f_{assign}(s_m^{ali})$ ▷ transfer the speakers from the source
13:         **else**
14:             $s_m^{tra} \leftarrow s_m^{tgt}$ ▷ preserve the target speaker if source speaker is unavailable
15:         **end if**
16:     **end for**
17: **end procedure**

Below we show a simple example of the inputs and output of the TPST algorithm:

Example inputs and output of the TPST algorithm

```
Source words:         hello good morning hi how are you pretty good
Source speakers:      1 1 1 2 2 2 2 1 1
Target words:         hello morning hi hey are you be good
Target speakers:      1 2 2 2 1 1 2 1
Transferred speakers: 1 1 2 2 2 2 1 1
```

3.5 LLM FINETUNING

Although the examples shown in Section 2 and Appendix B were using off-the-shelf Web APIs of commercial LLMs, finetuning the LLM specifically on the speaker diarization task is still required if we need to:

    1. Reduce errors of a specific speaker diarization system;

---

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

2. Handle more complicated errors;

3. Keep ASR transcripts unmodified as much as possible from the LLM outputs;

4. Avoid undesired leading or tailing text from the generated completions, such as "Here is the corrected transcript" or "We corrected the speakers for these words";

5. Use smaller and cheaper LLMs.

To finetune the LLM, we build our 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 will produce 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 our training data with three different flavors, as introduced below.

3.5.1 Flavor 1: hyp2ora

The first flavor is named *hypothesis-to-oracle*, or simply *hyp2ora*. In this flavor, we apply the Transcript-Preserving Speaker Transfer algorithm from Section 3.4 by treating reference sequences as source and hypothesis sequences as target:

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

where the output $\mathbf{s}^{ora}$ is the **oracle hypothesis speakers** transferred 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.

3.5.2 Flavor 2: deg2ref

The second flavor is named *degraded-to-reference*, or simply *deg2ref*. In this flavor, we apply the Transcript-Preserving Speaker Transfer algorithm from Section 3.4 by treating hypothesis sequences as source and reference sequences as target:

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

where the output $\mathbf{s}^{deg}$ is the **degraded reference speakers** transferred 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.

3.5.3 Flavor 3: mixed

The third flavor named *mixed* 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.

Note that 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.

4 EXPERIMENTS

4.1 DATASETS

To finetune the LLM, we use the training subset of the Fisher corpus [40], which consists of 1,920 hours of 11,527 conversations. The same train-test split of the Fisher dataset has been used in many previous works [8, 17, 35, 41]

For evaluation, we use the testing subset of the Fisher corpus [40], as well as the testing subset of Callhome American English data [42]. The Fisher testing subset consists of 28.7 hours of 172 conversations[3]. The Callhome American English testing subset consists of 1.7 hours of 20 conversations. Both datasets are in the telephone speech domain, and all conversations have 2 speakers.

4.2 METRICS

To evaluate the diarization performance, we use two metrics: the Word Diarization Error Rate (WDER) [25] and the concatenated minimum-permutation word error rate (cpWER) [43]. To briefly recap, WDER is defined as:

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

where,

1. S_{IS} is the number of ASR Substitutions with Incorrect Speaker tokens.
2. C_{IS} is the number of Correct ASR words with Incorrect Speaker tokens.
3. S is the number of ASR substitutions.
4. C is the number of Correct ASR words.

And 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 assumed to be the best permutation.

All three metrics reported in this paper (WER, WDER, and cpWER) are micro metrics, i.e. both numerators and denominators are aggregated on the entire dataset.

4.3 MODELS

For the ASR model in Fig. 2, we use a universal speech model (USM) [44] with 600 million parameters trained with the RNN-T loss [45]. For the speaker diarization model in Fig. 2, we use the turn-to-diarize system [7] with a multi-stage clustering setup [17] in our experiments, which is capable of diarizing hours of audio recordings in real time on a mobile device [24]. The number of speakers is unknown (from 1 to ∞) to the speaker diarization system in all of our experiments. Specifically, for the Fisher training set, the Fisher testing set, and the Callhome testing set, all utterances have a ground truth number of speakers equal to two, but our turn-to-diarize system may predict any number of hypothesis speakers. The histogram of number of hypothesis speakers predicted by turn-to-diarize on the Fisher training set is shown in Figure 3. Although our experiments are based on the USM + turn-to-diarize setup, we would like to point out that the proposed framework is very generic and should work with other ASR or speaker diarization systems as well, such as variants of end-to-end speaker diarization models [20, 21, 22, 23].

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 API) [46]. We use

---

[3]https://github.com/google/speaker-id/blob/master/publications/ScdLoss/eval/fisher.txt
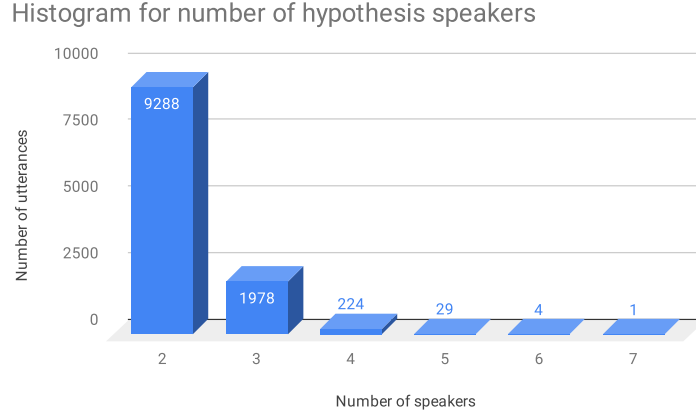
Figure 3: The histogram for the number of hypothesis speakers predicted by the turn-to-diarize system on the Fisher training set. Note that the ground truth number of speakers is always two on the Fisher dataset, but we do not constrain the number of speakers for the turn-to-diarize system.

the PaLM 2-S model as our 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 [47]. During finetuning, we limit the LLM input size by 4,096 tokens, and segment our training and testing data accordingly. The PaLM 2-L model will only be used for zero-shot and one-shot experiments, as described in Section 4.4. We also experimented with open source models such as Llama 2 [48] and Llama 3. Model details on how we finetune these models are provided in Appendix A.

In our prompt builder module, we use an empty prompt prefix, and a 5-character prompt suffix "␣-->␣" (note the two spaces around the arrow). For the completions in our 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 trained 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. This is more similar to the example we demonstrated in Section 2.

For the zero-shot setup, we use a prompt prefix that contains an instruction describing the task, as shown below.

> **Prompt prefix for zero-shot**
>
> 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.\n

For the one-shot setup, the prompt prefix contains both the instruction describing the task, and also a small example, as shown below.

9

Prompt prefix for one-shot

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.\n

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 the PaLM 2-S foundation model does not offer speaker diarization capabilities without finetuning. Zero-shot experiment with PaLM 2-L model also shows bad performance, while one-shot experiment with PaLM 2-L model is much better, but still worse than the baseline system. Our results indicate that the PaLM 2-L model with one-shot is able to improve speaker diarization in relatively simple cases as shown in Section 2 and Appendix B. However, real world applications can be much 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 improvement of both WDER and cpWER with any of the three finetuning flavors. Interesting, 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. Even if we did 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 demonstrated that the DiarizationLM solution provides consistent quality gains even with out-of-domain ASR and speaker diarization models.

To further demonstrate this, in Table 2, we show the results of a similar setup, but we replace the USM-based ASR model directly by the ground truth ASR transcripts from the testing sets. For these experiments, we will have WER=0%, and the hyp2ora and deg2ref flavors will be equivalent. From the table, we can still see big improvements of WDER after post-processing the diarization results by the same DiarizationLM model (i.e. deg2ref flavor in Table 1).

4.6   CASE STUDIES

Based on the results from Table 1, we also present example cases from the Fisher and Callhome testing sets where we see big improvements of WDER in Table 3 and Table 4, respectively. From these examples, we are seeing multiple patterns of corrections:

- DiarizationLM make corrections where **different parts of sentence** are moved to the same speaker, e.g. "it's more of" and "it'll be warm" in fe_03_07146 from Table 3. This is consistent with our initial observations as demonstrated in Section 2.

- DiarizationLM can merge short speaker turns due to **disfluency**, such as "yeah yeah" and "i i hear i hear " in fe_03_11159 from Table. 3. Diarization errors from disfluency usually attribute to low quality speaker embeddings extracted from very short speaker turn segments.

Table 1: Evaluation results of the USM + turn-to-diarize baseline system and the results post-processed by DiarizationLM. For DiarizationLM, we experiment with PaLM 2 foundation models with and without finetuning on the diarization task. We also experiment with Llama 2 and Llama 3 models finetuned on the diarization task (model details in Appendix A). WERs are the same for all systems due to TPST. All numbers are percentages.

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)	-	**2.37**	**16.93**	-	**4.25**	**20.22**
+ PaLM 2-S finetuned (deg2ref)	-	3.94	18.55	-	5.33	21.47
+ PaLM 2-S finetuned (mixed)	-	2.41	16.94	-	4.76	20.84
+ Llama 2 13B finetuned (mixed) v1	-	3.65	18.92	-	8.02	25.11
+ Llama 3 8B finetuned (mixed) v1	-	4.40	19.76	-	12.27	30.80
+ Llama 3 8B finetuned (mixed) v2	-	3.28	18.37	-	6.66	23.57

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. For DiarizationLM, we experiment with PaLM 2 foundation models with and without finetuning on the diarization task. All numbers are percentages.

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**

- DiarizationLM can also detect speaker turns due to **interruptions**, such as "oh all right" in `fe_03_11210` from Table 3, and "oh my" in `en_6408` from Table 4.

We also look into why zero-shot and one-shot experiments in Table 1 produced worse results than the baseline system. We found that without finetuning on the speaker diarization tasks, zero-shot and one-shot outputs from the LLM often delete big chunks of hypothesis text from the prompt. Finetuning the LLM is critical to avoid such undesired deletions. A few zero-shot examples with the PaLM 2-S model from the Fisher testing set were shown in Table 5.

5 DISCUSSION AND FUTURE WORK

The experiments in Section 4 have shown very promising results where LLMs can significantly reduce speaker diarization errors. However, we also admit the limitations of these experiments. First of all, the training and testing data from the experiments are all based on the telephone 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 unknown number of speakers.

In Appendix B, we have demonstrated other diarization capabilities of LLMs. However, due to lack of relevant datasets, we haven't been able to thoroughly evaluate these capabilities. One interesting future work would be to collect datasets of these tasks and evaluate how LLM performs.

Table 3: Example cases from the Fisher testing set where we see big absolute WDER reduction (Δ WDER) with DiarizationLM (deg2ref flavor).

Utterance	Before DiarizationLM	After DiarizationLM
fe_03_07146 (Δ WDER =8.80%)	... <spk:3> it's it's <spk:1> more of summer always like you know we never experience a bit cold over here <spk:4> usually it'll <spk:1> be warm or like very hot in summer yeah and <spk:3> extremely hot yeah with high humidity my humidity is pretty <spk:1> much high because i stay close to the sea coast over here <spk:3> yeah <spk:1> so <spk:3> that makes you live houston is it like houston where you live yeah i i i live <spk:1> in houston ...	... <spk:1> it's it's more of summer always like you know we never experience a bit cold over here usually it'll be warm or like very hot in summer <spk:2> yeah and extremely hot yeah with high humidity my <spk:1> humidity is pretty much high because i stay close to the sea coast over here <spk:2> yeah so that makes you live houston is it like houston where you live <spk:1> yeah i i i live in houston ...
fe_03_06816 (Δ WDER =6.61%)	... <spk:3> uhuh <spk:2> did you see the the woman golfer that was on this the one <spk:1> monica yeah yeah <spk:2> what's her name monica stone yeah mhm she she <spk:1> blew out she fell out of that tournament but i didn't think she'd do it she she's girls can't compete against guys ...	... <spk:2> uhuh did you see the the woman golfer that was on this the one <spk:1> monica yeah yeah <spk:2> what's her name monica stone <spk:1> yeah <spk:2> mhm <spk:1> she she blew out she fell out of that tournament but i didn't think she'd do it she she's girls can't compete against guys ...
fe_03_11210 (Δ WDER =6.35%)	... <spk:1> the vikings mine's the eagles i'm from new jersey oh all right i have my jersey on now i watch the game tonight yeah well i i may i may just watch <spk:2> part of it tonight too then but uh it's a case as i say if if i had to pay for it i probably wouldn't watch it <spk:1> i wouldn't either uhhuh <spk:2> unless <spk:1> it was an eagles game ...	... <spk:1> the vikings mine's the eagles i'm from new jersey <spk:2> oh all right <spk:1> i have my jersey on now i watch the game tonight yeah <spk:2> well i i may i may just watch part of it tonight too then but uh it's a case as i say if if i had to pay for it i probably wouldn't watch it <spk:1> i wouldn't either <spk:2> uhhuh <spk:1> unless it was an eagles game ...
fe_03_11159 (Δ WDER =4.05%)	... <spk:2> yeah <spk:1> anniversary that's horrible <spk:2> yeah <spk:1> yeah it's not good <spk:2> i <spk:1> i hear i hear you there that's not a good thing you <spk:2> know i mean of course you know that's a day that will go down instantly nobody will ever remember it ...	... <spk:1> yeah anniversary that's horrible yeah yeah it's not good i i hear i hear you there that's not a good thing <spk:2> you know i mean of course you know that's a day that will go down instantly nobody will ever remember it ...

Table 4: Example cases from the Callhome testing set where we see big absolute WDER reduction (Δ WDER) with DiarizationLM (deg2ref flavor).

Utterance	Before DiarizationLM	After DiarizationLM
en_6447 (Δ WDER =12.49%)	... <spk:1> i'm <spk:2> going to see if i can talk to the guy that's selling the trailer if i can chew him down a bit uhhuh <spk:1> and <spk:2> you know what you just said benedicta is are you living with benedicta <spk:1> yes yes yes <spk:2> you know what i bet she answered the phone ...	... <spk:2> i'm going to see if i can talk to the guy that's selling the trailer if i can chew him down a bit <spk:1> uhhuh <spk:2> and you know what you just said benedicta is are you living with benedicta <spk:1> yes yes yes <spk:2> you know what i bet she answered the phone ...
en_6408 (Δ WDER =10.87%)	... <spk:1> uhu <spk:2> so <spk:1> he had big surgery again and he's in a wheelchair oh my <spk:2> and <spk:1> he doesn't want to go to school in a wheelchair uhuh but <spk:2> he might he wants to have tutoring at home but they're still where they lived on 45th street <spk:1> yeah they're there ...	... <spk:2> uhu <spk:1> so he had big surgery again and he's in a wheelchair <spk:2> oh my <spk:1> and he doesn't want to go to school in a wheelchair <spk:2> uhuh <spk:1> but he might he wants to have tutoring at home <spk:2> but they're still where they lived on 45th street <spk:1> yeah they're there ...
en_6298 (Δ WDER =9.95%)	... <spk:1> um hey we're we're confused about you guys address <spk:2> is <spk:1> it 1324 or 13 <spk:2> it's 1 324 <spk:1> excuse me 1324 yes and it's me view is me two words or one word yes it's two words and there's an ln besides ...	... <spk:1> um hey we're we're confused about you guys address is it 1324 or 13 <spk:2> it's 1 324 <spk:1> excuse me 1324 <spk:2> yes <spk:1> and it's me view is me two words or one word <spk:2> yes it's two words and there's an ln besides ...
en_4792 (Δ WDER =9.42%)	... <spk:2> yeah well he was at columbia <spk:1> he was there like five years and they turned him down for tenure then he went somewhere else he he was down in college park maryland yeah and he i think he was only non tenure track down there then supposedly supposed to be back in japan now yeah but you know he's he's probably become an english teacher at some unit yeah i know a guy believe it or not i know a guy from manhattan who was up in sapotto his major he did an mba believe it or not he's he's an english teacher now huh ...	... <spk:2> yeah well he was at columbia <spk:1> he was there like five years and they turned him down for tenure then he went somewhere else he he was down in college park maryland <spk:2> yeah <spk:1> and he i think he was only non tenure track down there then supposedly supposed to be back in japan now <spk:2> yeah <spk:1> but you know he's he's probably become an english teacher at some unit <spk:2> yeah <spk:1> i know a guy believe it or not i know a guy from manhattan who was up in sapotto his major he did an mba believe it or not he's he's an english teacher now <spk:2> huh ...

Table 5: Example cases from the Fisher testing set where zero-shot PaLM 2-S deletes lots of text from the prompt.

Utterance	Before DiarizationLM	After DiarizationLM
fe_03_11252	... <spk:1> oh okay i believe it's a lot wrong with the public schools i don't believe that they're um that they're giving um these kids a sense of um well they're not teaching them what they need to know once they get out of um school you know mhm um what what's happening is that's probably why you got a lot of um a lot of people that's unemployed i think you know they you get a lot from school and they taking a lot of um i guess the economics out of school you know <spk:2> right ...	... <spk:1> oh okay i believe it's a lot wrong with the public schools i don't believe that they're um that they're giving um these kids a sense of um well they're not teaching them what they need to know once they get out of um school you know <spk:2> right ...
fe_03_11224	... <spk:1> so um i think what do you think is an important thing in a relation i think the topic was um what you um what are the most important things in a life partner yeah uh h well what do you think me <spk:2> i would have to say trust and honesty like cuz without that you really don't have nothing to build on you know right yeah ...	... <spk:1> so um i think what do you think is an important thing in a relation <spk:2> i would have to say trust and honesty like cuz without that you really don't have nothing to build on you know right ...

Another research direction would be to compare different LLMs, in 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 [20] or WEEND [34].

Lastly, as PaLM 2 models are multilingual [46], the DiarizationLM framework can naturally apply to speaker diarization tasks in other languages. It would be helpful to evaluate how DiarizationLM performs on speaker diarization datasets in other languages than English.

6 RELATED WORK

6.1 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 [49]. Later in [50], the authors proposed to use a neural network for resegmentation, with an additional class for non-speech. In [51], the authors proposed DiaCorrect, a method inspired by error correction techniques in ASR. DiaCorrect uses parallel convolutional encoders for the speakers from the initial diarization results and a transformer based decoder to produce corrected diarization results. 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 cluster level.

Another type of post-processing is to combine the outputs of multiple speaker diarization systems, e.g. via majority voting [52], speaker matching [53], or both [54]. More recently in [16], the authors proposed to perform speaker diarization on different temporal scales, and combine their outputs via 1-D convolutional neural networks. In [55], the authors proposed to use end-to-end speaker diarization as a post-processing step for initial speaker diarization results of a clustering-based system.

Our proposed framework is generic such that it can apply to either the results of a single speaker diarization system, or to the combined results of multiple speaker diarization systems.

6.2 SPEAKER DIARIZATION WITH SEMANTIC INFORMATION

Apart from the joint ASR and speaker diarization models discussed in Section 1, researchers have also studied various approaches of integrating semantic information into conventional speaker diarization systems. Some of the benefits of DiarizationLM may also be achieved with non-LLM methods.

The most common approach to leverage semantic information is to use ASR word alignments to refine the voice activity detection and initial segmentation [56]. A variant of this approach is to build a speaker turn detection model and segment by speaker turns [57]. In [58], a Gated Recurrent Units (GRUs) [59] based speaker turn probability estimator is trained on top of word embeddings and speaker embeddings, and the estimated probabilities are combined with the adjacency matrix for spectral clustering. Similarly in [7], an end-to-end trained transformer transducer (T-T) [60] based speaker turn detection model is used to constrain the spectral clustering via Exhaustive and Efficient Constraint Propagation (E2CP).

6.3 SPEAKER DIARIZATION WITH LLM

In [35], the authors proposed Speaker Error Corrector (SEC), which aims to solve the same problem as we stated in Section 1. In [35], word embeddings from the ASR transcript are extracted with a pre-trained Roberta-base LM [61]. Then a separately trained transformer encoder takes the word embeddings and the hypothesis 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 from our proposed framework to [35] 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. Our DiarizationLM is a "text-in, text-out" system, without relying on internal embedding representations from the LLM.

More recently in [62], the authors proposed to use LLM to predict the speaker probability for the next word, and incorporate this probability into the beam search decoding of speaker diarization. Our proposed framework differs from this work by using a single prompt (or several prompts due to LLM input size limit) to post-process the entire results of the speaker diarization system, instead of word-by-word prompting. Additionally, our proposed framework can be more generally applied to any speaker diarization system, instead of requiring word-level speaker probabilities for beam search decoding.

7 CONCLUSION

In this paper, we demonstrate that large language models (LLM) can be used to post-process speaker diarization results, achieving various goals such as improving the readability of the diarization transcript, and reducing the diarization errors. Specifically, we proposed DiarizationLM, a framework where we use a finetuned LLM to refine the results from off-the-shelf ASR and speaker diarization systems. We introduced three different flavors to build the prompt-completion pairs data for finetuning the LLM. Our experiments on Fisher and Callhome datasets show that a finetuned PaLM 2-S model can drastically reduce the word diarization error rates of typical diarization systems like turn-to-diarize.

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," *Computer Speech & Language*, vol. 72, pp. 101317, 2022.

[2] Chao Zhang and Quan Wang, "Speaker diarization: A journey from unsupervised to supervised approaches," Odyssey: The Speaker and Language Recognition Workshop, 2022, Tutorial session.

[3] 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.

[4] Ivan Medennikov, Maxim Korenevsky, Tatiana Prisyach, Yuri Khokhlov, Mariya Korenevskaya, Ivan Sorokin, Tatiana Timofeeva, Anton Mitrofanov, Andrei Andrusenko, Ivan Podluzhny, et al., "Target-speaker voice activity detection: a novel approach for multi-speaker diarization in a dinner party scenario," in *Proc. Interspeech*, 2020.

[5] Shaojin Ding, Quan Wang, Shuo-yiin Chang, Li Wan, and Ignacio Lopez Moreno, "Personal VAD: Speaker-conditioned voice activity detection," in *Odyssey: The Speaker and Language Recognition Workshop*, 2020.

[6] Shaojin Ding, Rajeev Rikhye, Qiao Liang, Yanzhang He, Quan Wang, Arun Narayanan, Tom O'Malley, and Ian McGraw, "Personal vad 2.0: Optimizing personal voice activity detection for on-device speech recognition," *arXiv preprint arXiv:2204.03793*, 2022.

[7] Wei Xia, Han Lu, Quan Wang, Anshuman Tripathi, Yiling Huang, Ignacio Lopez Moreno, and Hasim Sak, "Turn-to-Diarize: Online speaker diarization constrained by transformer transducer speaker turn detection," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8077–8081.

[8] Guanlong Zhao, Quan Wang, Han Lu, Yiling Huang, and Ignacio Lopez Moreno, "Augmenting transformer-transducer based speaker change detection with token-level training loss," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[9] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[10] 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*, 2017.

[11] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-Vectors: Robust dnn embeddings for speaker recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[12] Quan Wang, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopez Moreno, "Speaker diarization with LSTM," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5239–5243.

[13] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree, "Speaker diarization using deep neural network embeddings," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4930–4934.

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

[15] 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.

[16] Tae Jin Park, Manoj Kumar, and Shrikanth Narayanan, "Multi-scale speaker diarization with neural affinity score fusion," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7173–7177.

[17] Quan Wang, Yiling Huang, Han Lu, Guanlong Zhao, and Ignacio Lopez Moreno, "Highly efficient real-time streaming and fully on-device speaker diarization with multi-stage clustering," *arXiv preprint arXiv:2210.13690*, 2022.

[18] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang, "Fully supervised speaker diarization," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6301–6305.

[19] 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.

[20] 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.

[21] Yushi Ueda, Soumi Maiti, Shinji Watanabe, Chunlei Zhang, Meng Yu, Shi-Xiong Zhang, and Yong Xu, "EEND-SS: Joint end-to-end neural speaker diarization and speech separation for flexible number of speakers," *arXiv preprint arXiv:2203.17068*, 2022.

[22] Quan Wang, Yash Sheth, Ignacio Lopez Moreno, and Li Wan, "Speaker diarization using an end-to-end model," US Patent US011545157B2, 2019.

[23] Shota Horiguchi, Yusuke Fujita, Shinji Watanabe, Yawen Xue, and Kenji Nagamatsu, "End-to-end speaker diarization for an unknown number of speakers with encoder-decoder based attractors," *arXiv preprint arXiv:2005.09921*, 2020.

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

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

[26] Naoyuki Kanda, Yashesh Gaur, Xiaofei Wang, Zhong Meng, Zhuo Chen, Tianyan Zhou, and Takuya Yoshioka, "Joint speaker counting, speech recognition, and speaker identification for overlapped speech of any number of speakers," *arXiv preprint arXiv:2006.10930*, 2020.

[27] Naoyuki Kanda, Zhong Meng, Liang Lu, Yashesh Gaur, Xiaofei Wang, Zhuo Chen, and Takuya Yoshioka, "Minimum bayes risk training for end-to-end speaker-attributed ASR," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6503–6507.

[28] Naoyuki Kanda, Guoli Ye, Yashesh Gaur, Xiaofei Wang, Zhong Meng, Zhuo Chen, and Takuya Yoshioka, "End-to-end speaker-attributed ASR with transformer," *arXiv preprint arXiv:2104.02128*, 2021.

[29] Naoyuki Kanda, Jian Wu, Yu Wu, Xiong Xiao, Zhong Meng, Xiaofei Wang, Yashesh Gaur, Zhuo Chen, Jinyu Li, and Takuya Yoshioka, "Streaming speaker-attributed ASR with token-level speaker embeddings," *arXiv preprint arXiv:2203.16685*, 2022.

[30] Katerina Zmolikova, Marc Delcroix, Keisuke Kinoshita, Takuya Higuchi, Atsunori Ogawa, and Tomohiro Nakatani, "Speaker-aware neural network based beamformer for speaker extraction in speech mixtures," in *Proc. Interspeech*, 2017, pp. 2655–2659.

[31] Marc Delcroix, Katerina Zmolikova, Keisuke Kinoshita, Atsunori Ogawa, and Tomohiro Nakatani, "Single channel target speaker extraction and recognition with speaker beam," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5554–5558.

[32] Marc Delcroix, Shinji Watanabe, Tsubasa Ochiai, Keisuke Kinoshita, Shigeki Karita, Atsunori Ogawa, and Tomohiro Nakatani, "End-to-end SpeakerBeam for single channel target speech recognition," in *Proc. Interspeech*, 2019, pp. 451–455.

[33] Naoyuki Kanda, Shota Horiguchi, Ryoichi Takashima, Yusuke Fujita, Kenji Nagamatsu, and Shinji Watanabe, "Auxiliary interference speaker loss for target-speaker speech recognition," *arXiv preprint arXiv:1906.10876*, 2019.

[34] Yiling Huang, Weiran Wang, Guanlong Zhao, Hank Liao, Wei Xia, and Quan Wang, "On the success and limitations of auxiliary network based word-level end-to-end neural speaker diarization," in *Proc. Interspeech*, 2024, pp. 32–36.

[35] Rohit Paturi, Sundararajan Srinivasan, and Xiang Li, "Lexical speaker error correction: Leveraging language models for speaker diarization error correction," *arXiv preprint arXiv:2306.09313*, 2023.

[36] James Manyika and Sissie Hsiao, "An overview of Bard: an early experiment with generative AI," https://ai.google/static/documents/google-about-bard.pdf, 2023.

[37] OpenAI, "Introducing ChatGPT," https://openai.com/blog/chatgpt, 2022.

[38] Vladimir I Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[39] Harold W Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[40] 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.

[41] Guanlong Zhao, Yongqiang Wang, Jason Pelecanos, Yu Zhang, Hank Liao, Yiling Huang, Han Lu, and Quan Wang, "USM-SCD: Multilingual speaker change detection based on large pretrained foundation models," *arXiv preprint arXiv:2309.08023*, 2023.

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

[43] Shinji Watanabe, Michael Mandel, Jon Barker, Emmanuel Vincent, Ashish Arora, Xuankai Chang, Sanjeev Khudanpur, Vimal Manohar, Daniel Povey, Desh Raj, et al., "CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings," *arXiv preprint arXiv:2004.09249*, 2020.

[44] Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, et al., "Google USM: Scaling automatic speech recognition beyond 100 languages," *arXiv preprint arXiv:2303.01037*, 2023.

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

[46] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al., "PaLM 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[47] Taku Kudo and John Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[49] Gregory Sell and Daniel Garcia-Romero, "Diarization resegmentation in the factor analysis subspace," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4794–4798.

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

[51] Jiangyu Han, Federico Landini, Johan Rohdin, Mireia Diez, Lukas Burget, Yuhang Cao, Heng Lu, and Jan Cernocky, "DiaCorrect: Error correction back-end for speaker diarization," *arXiv preprint arXiv:2309.08377*, 2023.

[52] MAH Huijbregts, David A van Leeuwen, and FM Jong, "The majority wins: a method for combining speaker diarization systems," in *Proc. Interspeech*, 2009.

[53] Simon Bozonnet, Nicholas Evans, Xavier Anguera, Oriol Vinyals, Gerald Friedland, and Corinne Fredouille, "System output combination for improved speaker diarization," in *Proc. Interspeech*, 2010.

[54] Andreas Stolcke and Takuya Yoshioka, "DOVER: A method for combining diarization outputs," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 757–763.

[55] Shota Horiguchi, Paola Garcia, Yusuke Fujita, Shinji Watanabe, and Kenji Nagamatsu, "End-to-end speaker diarization as post-processing," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7188–7192.

[56] Jan Silovsky, Jindrich Zdansky, Jan Nouza, Petr Cerva, and Jan Prazak, "Incorporation of the ASR output in speaker segmentation and clustering within the task of speaker diarization of broadcast streams," in *International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2012, pp. 118–123.

[57] Tae Jin Park and Panayiotis Georgiou, "Multimodal speaker segmentation and diarization using lexical and acoustic cues via sequence to sequence neural networks," in *Proc. Interspeech*, 2018, pp. 1373–1377.

[58] Tae Jin Park, Kyu J Han, Jing Huang, Xiaodong He, Bowen Zhou, Panayiotis Georgiou, and Shrikanth Narayanan, "Speaker diarization with lexical information," *arXiv preprint arXiv:2004.06756*, 2020.

[59] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[60] 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 *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

[61] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[62] Tae Jin Park, Kunal Dhawan, Nithin Koluguri, and Jagadeesh Balam, "Enhancing speaker diarization with large language models: A contextual beam search approach," *arXiv preprint arXiv:2309.05248*, 2023.

[63] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

Appendices

A   OPEN SOURCE MODELS

Apart from PaLM 2 [46] models, we have also experimented with open source models such as Llama 2 [48] and Llama 3. These finetuned models are publicly available on Hugging Face. The finetuning scripts are available at: `https://github.com/google/speaker-id/tree/master/DiarizationLM/unsloth`.

A.1   `google/DiarizationLM-13b-Fisher-v1`

Model URL:	`https://huggingface.co/google/DiarizationLM-13b-Fisher-v1`
Foundation model:	`https://huggingface.co/unsloth/llama-2-13b-bnb-4bit`

This model is finetuned on the training subset of the Fisher corpus, using a LoRA adapter [63] of rank 256. The total number of training parameters is 1,001,390,080. With a batch size of 16, this model has been trained for 12000 steps, which is ∼4 epochs of the training data.

We use the "mixed" flavor during our training, meaning we combine data from "hyp2ora" and "deg2ref" flavors (see Section 3.5 for the definition of these flavors). After the prompt builder, we have a total of 48,142 prompt-completion pairs in our training set.

The finetuning took more than 3 days on a Google Cloud VM instance that has one NVIDIA A100 GPU with 80GB memory.

The maximal length of the prompt to this model is 6000 characters, including the "␣-->␣" suffix. The maximal sequence length is 4096 tokens.

The evaluation results of this model is also provided in Table 1. We can see that similar to the finetuned PaLM 2-S models, this model also significantly reduces the WDER and cpWER on the Fisher testing set. However, on the Callhome testing set, we are seeing degradation in WDER and cpWER, indicating overfitting on the Fisher dataset. We have found this is because the loss is computed on both the prompt and the completion tokens during training.

A.2   `google/DiarizationLM-8b-Fisher-v1`

Model URL:	`https://huggingface.co/google/DiarizationLM-8b-Fisher-v1`
Foundation model:	`https://huggingface.co/unsloth/llama-3-8b-bnb-4bit`

This model is finetuned on the training subset of the Fisher corpus, using a LoRA adapter [63] of rank 256. The total number of training parameters is 671,088,640. With a batch size of 16, this model has been trained for 25400 steps, which is ∼8 epochs of the training data.

Other configurations of this model is the same as the model in the previous section. Note that we still use a maximal sequence length of 4096 tokens even if Llama 3 supports longer sequence.

The finetuning took more than 4 days on a Google Cloud VM instance that has one NVIDIA A100 GPU with 80GB memory.

The evaluation results of this model is also provided in Table 1. We can see that its performance is significantly worse than the Llama 2 13B model from Appendix A.1, but still better than the USM + turn-to-diarize baseline on Fisher testing set. Similarly, the performance on Callhome is worse than baseline, because the loss is computed on both the prompt and the completion tokens during training.

A.3 `google/DiarizationLM-8b-Fisher-v2`

Model URL:	https://huggingface.co/google/ DiarizationLM-8b-Fisher-v1
Foundation model:	https://huggingface.co/unsloth/ llama-3-8b-bnb-4bit

This model is finetuned on the training subset of the Fisher corpus, using a LoRA adapter [63] of rank 256. The total number of training parameters is 671,088,640. With a batch size of 16, this model has been trained for 28800 steps, which is ∼9 epochs of the training data.

Note that when training this model, **the loss is only computed on the completion tokens**. This is the biggest different between this model and the previous model in Appendix A.2 (`google/DiarizationLM-8b-Fisher-v1`).

Other configurations of this model is the same as the model in the previous section. Note that we still use a maximal sequence length of 4096 tokens even if Llama 3 supports longer sequence.

The finetuning took more than 4 days on a Google Cloud VM instance that has one NVIDIA A100 GPU with 80GB memory.

The evaluation results of this model is also provided in Table 1. We can see that its performance is significantly better than the model from Appendix A.2. Specifically, we no longer see degradation on Callhome testin set.

B OTHER DIARIZATION CAPABILITIES OF LLMS

In our studies, we have found multiple diarization capabilities of LLMs. Due to the lack of relevant training and testing datasets, this paper mainly focuses on evaluating how LLMs can reduce diarization errors. In this section, we provide several examples to demonstrate LLM's other diarization capabilities. All the provided examples are obtained by directly sending the prompts to Google Bard [36][4].

B.1 AUTOFILLING SPEAKER NAMES

In real world applications, real speaker names or aliases are much more readable than anonymous speaker labels such as "speaker 1" or "speaker 2". In many cases, speaker names can be obtained from the context of the conversation and from additional metadata of the conversation. In meeting summarization, the names can be obtained from the list of meeting attendants. And for video captioning, the names may be extracted from the video title or description. An example is provided below.

Prompt:

[video title] Tom meets Patrick in the parking lot
[video description] A short conversation

Replace the <spk:?> tokens from the below video caption with actual person names.

<spk:1> Good morning Patrick, how are you?
<spk:2> Good, good. How are you Tom?
<spk:1> Pretty good. Going to work?
<spk:2> Yes. Busy day. How are your kids? Do they go to school?
<spk:1> Oh they are too young for that. I sent them to daycare earlier today.
<spk:2> Oh yeah I forgot about that.

Completion:

---

[4]We used an internal version of Bard that is based on a larger model and supports more tokens than the public version.

> Based on the video title "Tom meets Patrick in the parking lot", here's the transcript with speaker names replaced:
>
> Tom: Good morning Patrick, how are you?
> Patrick: Good, good. How are you Tom?
> Tom: Pretty good. Going to work?
> Patrick: Yes. Busy day. How are your kids? Do they go to school?
> Tom: Oh they are too young for that. I sent them to daycare earlier today.
> Patrick: Oh yeah, I forgot about that.

B.2 AUTOFILLING SPEAKER ROLES

In some applications, knowing the roles of each speaker is useful enough. For example, it would be helpful to distinguish the doctor from the patient in medical transcripts [25], and to distinguish the customer from the agent in call center conversations. This task can be handled with LLM usually without additional metadata about the conversation. We show an example below.

> **Prompt:**
>
> Replace the <spk:?> tokens from the below conversation with roles like teacher, student, doctor, patient, etc.
>
> <spk:1> Hi, how can I help you today?
> <spk:2> Hi, I recently often feel quite dizzy at work.
> <spk:1> Do you have any of these symptoms? Like coughing, fever, or running nose?
> <spk:2> No, I don't think so.
> <spk:1> What do you usually eat for breakfast, lunch and dinner?
> <spk:2> I'm on a diet, so not eating very much. Mostly just one slice of toast in the morning, and eating salad for lunch and dinner.
>
> **Completion:**
>
> Doctor: Hi, how can I help you today?
> Patient: Hi, I recently often feel quite dizzy at work.
> Doctor: Do you have any of these symptoms? Like coughing, fever, or running nose?
> Patient: No, I don't think so.
> Doctor: What do you usually eat for breakfast, lunch, and dinner?
> Patient: I'm on a diet, so not eating very much. Mostly just one slice of toast in the morning, and eating salad for lunch and dinner.

B.3 REPLACING THE ORCHESTRATION MODULE

In Fig. 1, we have explained how an orchestration module can combine the ASR transcripts with speaker diarization outputs via the timing information from the two systems. Interestingly, we have found that this process can also be fully replaced by LLM. This can be achieved by explicitly including the timing information in the textual representation of both ASR transcripts and speaker diarization outputs. Additionally, more prompt engineering will be needed, such as explicitly explaining the format of the textual representation, and providing a one-shot example in the prompt. We show an example below.

> **Prompt:**
>
> Here we define the problem of speaker-transcript alignment. The transcript is represented by multiple entries of text, where each text has a starting time and an ending time. The speaker is also represented in this format. The alignment problem will assign a speaker to each word in the text, based on which speaker overlaps the most with that word in time.

Below is an example.

Transcript represented in format "[start - end] text":
[0 - 2.3] Good morning Patrick
[2.5 - 5.2] how are you?
[5.6 - 6.1] Good, good.
[6.2 - 8.3] How are you Tom?
[9.2 - 9.9] Pretty good.
[10.0 - 11.1] Going to work?
[12.5 - 13.6] Yes. Busy day.

Speaker represented in format "[start - end] speaker":
[0 - 5.1] <spk:1>
[5.3 - 8.7] <spk:2>
[9.2 - 10.9] <spk:1>
[12.1 - 13.5] <spk:2>

The alignment result should be:
[0 - 2.3] <spk:1> Good morning Patrick
[2.5 - 5.2] <spk:1> how are you?
[5.6 - 6.1] <spk:2> Good, good.
[6.2 - 8.3] <spk:2> How are you Tom?
[9.2 - 9.9] <spk:1> Pretty good.
[10.0 - 11.1] <spk:1> Going to work?
[12.5 - 13.6] <spk:2> Yes. Busy day.

Based on the example above, we solve the alignment problem below.

Transcript represented in format "[start - end] text":
[0 - 5.6] Hi, how can I help you today?
[6.2 - 11.1] Hi, I recently often feel quite dizzy at work.
[11.6 - 15.5] Do you have any of these symptoms?
[16.6 - 18.5] Like coughing, fever, or running nose?
[20.0 - 21.1] No, I don't think so.
[22.2 - 29.9] What do you usually eat for breakfast, lunch and dinner?
[31.2 - 34.8] I'm on a diet, so not eating very much.
[35.2 - 39.8] Mostly just one slice of toast in the morning, and eating salad for lunch and dinner.

Speaker represented in format "[start - end] speaker":
[0.3 - 5.3] <spk:1>
[6.0 - 12.0] <spk:2>
[12.9 - 20.1] <spk:1>
[20.2 - 21.0] <spk:2>
[21.8 - 31.1] <spk:1>
[32.4 - 40.7] <spk:2>

The alignment result should be:

Completion:

Here's the aligned transcript:

[0 - 5.6] <spk:1> Hi, how can I help you today?
[6.2 - 11.1] <spk:2> Hi, I recently often feel quite dizzy at work.
[11.6 - 15.5] <spk:1> Do you have any of these symptoms?
[16.6 - 18.5] <spk:1> Like coughing, fever, or running nose?
[20.0 - 21.1] <spk:2> No, I don't think so.

[22.2 - 29.9] <spk:1> What do you usually eat for breakfast, lunch and dinner?
[31.2 - 34.8] <spk:2> I'm on a diet, so not eating very much.
[35.2 - 39.8] <spk:2> Mostly just one slice of toast in the morning, and eating salad for lunch and dinner.

To align the transcript and speaker segments, we assign each word in the transcript to the speaker whose segment overlaps the most with that word in time.