The Speaker and Language Recognition Workshop (Odyssey 2024)
18-21 June 2024, Quebec City, Canada

On Speaker Attribution with SURT

Desh Raj[1], *Matthew Wiesner*[2], *Matthew Maciejewski*[2], *Leibny Paola García-Perera*[1,2], *Daniel Povey*[3], *Sanjeev Khudanpur*[1,2]

[1]CLSP & [2]HLTCOE, Johns Hopkins University, Baltimore, USA; [3]Xiaomi Corp., Beijing, China

draj@cs.jhu.edu, {wiesner, mmaciej2, leibny}@jhu.edu, dpovey@gmail.com, khudanpur@jhu.edu

Abstract

The Streaming Unmixing and Recognition Transducer (SURT) has recently become a popular framework for continuous, streaming, multi-talker speech recognition (ASR). With advances in architecture, objectives, and mixture simulation methods, it was demonstrated that SURT can be an efficient streaming method for *speaker-agnostic* transcription of real meetings. In this work, we push this framework further by proposing methods to perform *speaker-attributed* transcription with SURT, for both short mixtures and long recordings. We achieve this by adding an auxiliary speaker branch to SURT, and synchronizing its label prediction with ASR token prediction through HAT-style blank factorization. In order to ensure consistency in relative speaker labels across different utterance groups in a recording, we propose "speaker prefixing" — appending each chunk with high-confidence frames of speakers identified in previous chunks, to establish the relative order. We perform extensive ablation experiments on synthetic LibriSpeech mixtures to validate our design choices, and demonstrate the efficacy of our final model on the AMI corpus.

Index Terms: multi-talker ASR, SURT, speaker attribution, meeting transcription.

1. Introduction

Speaker-attributed multi-talker speech recognition (ASR), or "who spoke what", is the task of transcribing all the speech in a multi-talker conversation along with relative speaker attribution. This task has several applications such as meeting transcription and summarization, collaborative learning, and dinner-party conversations [1, 2, 3]. Due to the presence of overlapping speech, turn-taking, and far-field audio, it often requires special modeling techniques [4, 5, 6]. Researchers have worked on speaker-attributed transcription from modular (i.e., pipeline-based) and end-to-end perspectives. In the former, it is decomposed into speaker diarization and ASR sub-tasks and addressed independently, leveraging advances in each of these fields [7, 8, 9]. However, this approach may be sub-optimal since the components are independently optimized leading to error propagation, and may also require greater engineering efforts for maintenance [10].

Due to these limitations with modular systems, researchers have proposed jointly optimized models that combine diarization and ASR to directly solve for speaker-attributed transcription. The most popular of these is speaker-attributed ASR (SA-ASR) based on attention-based encoder-decoders (AEDs) [11]. It uses serialized output training (SOT) to handle overlapped speech and registered speaker profiles (called a speaker inventory) to handle speaker attribution [12, 13]. Several modifications to this model have leveraged transformer-based encoders [14] and large-scale pre-training [15], and have proposed methods for inference on long recordings [16] without the dependence on speaker inventory [17]. There have been further investigations on methods for speaker attribution within SA-ASR, and its extension to multi-channel and contextualized ASR [18, 19, 20]. By modifying SOT to be performed at the token-level (known as t-SOT), Kanda et al. [21] performed streaming transcription of overlapping speech, which was not feasible with utterance-level serialization. Enforcing monotonicity in this manner also allows these models to be built upon neural transducers [22] instead of AEDs. Nevertheless, t-SOT requires complicated interleaving/deserialization of tokens based on timestamps to accommodate overlapping speech on a single output channel, and the use of "channel change" tokens may impact ASR training adversely.

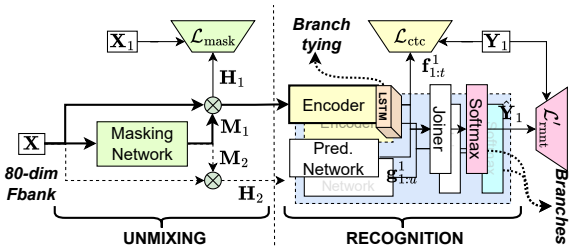


Figure 1: An overview of SURT 2.0, as described in [23]. It consists of a masking network and a transducer-based ASR.

An alternative approach for continuous, streaming, multi-talker ASR involves transcribing overlapping utterances on parallel output channels by unmixing them inside the model. This two-branch strategy is exemplified by models such as Streaming Unmixing and Recognition Transducer (SURT) [24] and multi-turn RNN-T (MT-RNNT) [25], but we will refer to them as SURT in this paper without loss of generality. SURT has been extended to handle long-form multi-turn recordings [26, 25], and to jointly perform endpointing and segmentation [27, 28]. Lu et al. [29] also proposed joint speaker identification with SURT, but their model relied on a speaker inventory and was only used for single-turn synthetic mixtures. As shown in Fig. 1, the SURT model consists of an "unmixing" component that separates the mixed audio into non-overlapping streams, and a "recognition" component that transcribes each of these streams. Since there is no explicit emission of speaker labels in this modeling scheme, SURT has thus far been limited to *speaker-agnostic* transcription. In this paper, our objective is to extend the SURT model for *speaker-attributed* transcription of an arbi-

This work was partially funded by the National Science Foundation CCRI program via Grant No. 2120435.

10.21437/odyssey.2024-14

trary number of speakers without any speaker inventory.

We achieve this by adding an auxiliary speaker transducer to the recognition module of SURT. We constrain this branch to emit a speaker label corresponding to each ASR token predicted by using blank factorization of the output logits, similar to the hybrid autoregressive transducer (HAT) model [30]. We also propose a novel "speaker prefixing" method to ensure that the speaker labels are consistent across different utterance groups in the recording. We validate our methods through ablation experiments on LibriSpeech mixtures, and finally demonstrate streaming speaker-attributed transcription on real meetings from the AMI corpus. Code is released through the open-source `icefall` toolkit: `https://github.com/k2-fsa/icefall`.

2. Preliminary: SURT

2.1. Speech recognition with neural transducers

In single-talker ASR, audio features for a segmented utterance $\mathbf{X} \in \mathbb{R}^{T \times F}$ (T and F denote the number of time frames and the input feature dimension, respectively) are provided as input, and the system predicts the transcript $\mathbf{y} = (y_1, \ldots, y_U)$, where $y_u \in \mathcal{V}$ are output units such as graphemes or word-pieces, and U is the length of the label sequence. For discriminative training, we achieve this by minimizing the negative conditional log-likelihood, $\mathcal{L} = -\log P(\mathbf{y}|\mathbf{X})$. Since the alignment between $\mathbf{X}$ and $\mathbf{y}$ is not known, transducers compute $\mathcal{L}$ by marginalizing over the set of all alignments $\mathbf{a} \in \bar{\mathcal{V}}^{T+U}$, where $\bar{\mathcal{V}} = \mathcal{V} \cup \{\phi\}$ and ϕ is called the blank label. Formally,

$$P(\mathbf{y}|\mathbf{X}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} P(\mathbf{a}|\mathbf{X}), \tag{1}$$

where $\mathcal{B}$ is a deterministic mapping from an alignment $\mathbf{a}$ to an output sequence $\mathbf{y}$ that removes the blank labels. Transducers parameterize $P(\mathbf{a}|\mathbf{X})$ with an encoder, a prediction network, and a joiner (see "recognition" component in Fig. 1). The encoder maps $\mathbf{X}$ into hidden representations $\mathbf{f}_{1:T}$, while the prediction network auto-regressively maps $\mathbf{y}$ into $\mathbf{g}_{1:U}$. The joiner combines the outputs from the encoder and the prediction network to compute logits $\mathbf{Z} \in \mathbb{R}^{T \times U \times |\bar{\mathcal{V}}|}$ which are fed to a softmax function to produce a posterior distribution over $\bar{\mathcal{V}}$ for each (t, u). Under the assumption of a streaming encoder, we can expand (1) as

$$P(\mathbf{y}|\mathbf{X}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{t=1}^{T+U} P(\mathbf{a}_t|\mathbf{f}_{1:t}, \mathbf{g}_{1:u(t-1)}) \tag{2}$$

$$= \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{t=1}^{T+U} \text{Softmax}(\mathbf{z}_{t,u(t)}), \tag{3}$$

where $u(t) \in \{1, \ldots, U\}$ denotes the index in the label sequence at time t, and $\mathbf{z}_{t,u}$ denotes the logits over $\bar{\mathcal{V}}$ for each (t, u). The negative log of this expression is known as the RNN-T or transducer loss. In practice, to make training more memory-efficient, we often approximate the full sum, for example using the pruned transducer loss [31]. We will denote this loss as $\mathcal{L}_{\text{rnnt}}$ for the remainder of this paper.

2.2. Multi-talker ASR with SURT

In multi-talker ASR, the input $\mathbf{X} \in \mathbb{R}^{T \times F}$ is an unsegmented mixture containing N utterances from K speakers, i.e., $\mathbf{X} = \sum_{n=1}^{N} \mathbf{x}_n$, where $\mathbf{x}_n$ is the n-th utterance ordered by start time, shifted and zero-padded to the length of $\mathbf{X}$. The desired output is $\mathbf{Y} = \{\mathbf{y}_n : 1 \leq n \leq N\}$, where $\mathbf{y}_n$ is the reference corresponding to $\mathbf{x}_n$. Assuming at most two-speaker overlap, the *heuristic error assignment training* (HEAT) paradigm [24] is used to create channel-wise references $\mathbf{Y}_1$ and $\mathbf{Y}_2$ by assigning $\mathbf{y}_n$'s to the first available channel, in order of start time. SURT estimates $\hat{\mathbf{Y}} = [\hat{\mathbf{Y}}_1, \hat{\mathbf{Y}}_2] = f_{\text{surt}}(\mathbf{X})$ as follows. First, an unmixing module computes $\mathbf{H}_1$ and $\mathbf{H}_2$ as

$$\begin{aligned} &\mathbf{H}_1 = \mathbf{M}_1 * \mathbf{X}, \quad \mathbf{H}_2 = \mathbf{M}_2 * \mathbf{X}, \quad \text{where} \\ &[\mathbf{M}_1, \mathbf{M}_2]^T = \text{MaskNet}(\mathbf{X}), \end{aligned} \tag{4}$$

$\mathbf{M}_c \in \mathbb{R}^{T \times F}$ is a soft mask per channel and $*$ is Hadamard product. $\mathbf{H}_1$ and $\mathbf{H}_2$ are fed into a transducer-based ASR, producing logits $\mathbf{Z}_1$ and $\mathbf{Z}_2$. Finally,

$$\begin{aligned} &\mathcal{L}_{\text{heat}} = \mathcal{L}(\mathbf{X}, \mathbf{Y}_1, \mathbf{Z}_1) + \mathcal{L}(\mathbf{X}, \mathbf{Y}_2, \mathbf{Z}_2), \quad \text{where} \\ &\mathcal{L} = \mathcal{L}_{\text{rnnt}} + \lambda_{\text{ctc}} \mathcal{L}_{\text{ctc}} + \lambda_{\text{mask}} \mathcal{L}_{\text{mask}}, \end{aligned} \tag{5}$$

where $\mathcal{L}_{\text{ctc}}$ and $\mathcal{L}_{\text{mask}}$ denote auxiliary CTC loss on the encoder [32] and mean-squared error loss on the masking network, respectively, and λ's are hyperparameters. In the original formulation, SURT only performs *speaker-agnostic* transcription, and is evaluated using ORC-WER (Section 4.4) [25].

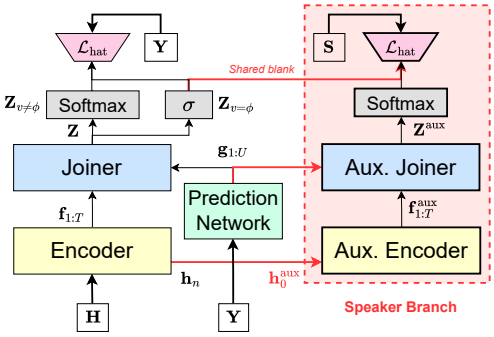


Figure 2: Auxiliary speaker transducer (red box) with shared blank label. The auxiliary encoder takes as input a hidden layer representation from the main encoder (i.e., $\mathbf{h}^{\text{aux}}_0 = \mathbf{h}_n$), and generates $\mathbf{f}^{\text{aux}}_{1:T}$. The blank logit $\mathbf{Z}_{v=\phi}$ from the main joiner is shared with the speaker branch to compute the HAT loss. Note that here we show only one of the two branches/channels of the SURT recognition module.

3. Methodology

For speaker-attributed ASR, the desired output is $\mathbf{Y} = \{(\mathbf{y}_n, s_n) : 1 \leq n \leq N, s_n \in [1, K]\}$, where K is the number of speakers in the mixture. SURT estimates $\mathbf{y}_n$ by mapping the utterances to two channels $\hat{\mathbf{Y}}_1$ and $\hat{\mathbf{Y}}_2$, as described in Section 2.2. A popular method for speaker attribution in multi-talker settings is to predict speaker change tokens that segment the output into speaker-specific regions, followed by speaker label assignment to each segment. However, this kind of training is prone to over-estimate the speaker change tokens, and may also adversely affect the ASR performance. Instead, we want to perform speaker attribution without affecting the output of the ASR branch, for example by predicting a speaker label for each ASR token emitted.

In order to perform such a streaming speaker attribution jointly with the transcription, the following questions arise: (i) How do we deal with overlapping speech? (ii) How do we synchronize speaker label prediction with ASR token prediction? (iii) How to reconcile relative speaker labels across utterance groups in a long recording? We will answer each of these questions in the following subsections.

3.1. Auxiliary speaker transducer

We map speaker labels s_n to two channels according to the HEAT strategy, obtaining $\hat{\mathbf{S}}_1$ and $\hat{\mathbf{S}}_2$. During training, we

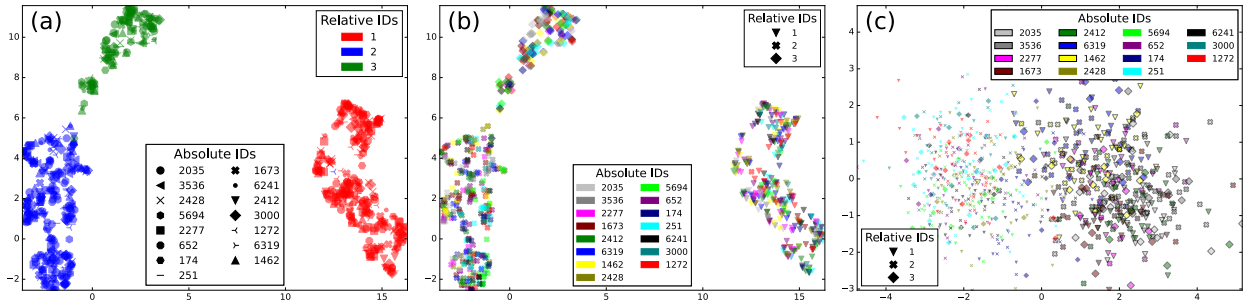


Figure 3: Projections of auxiliary encoder representations for a subset of LSMix `dev`. Each point denotes the representation of one speaker in a mixture, averaged over the frames on which the model emits a non-blank label. (a) and (b) denote UMAP projection, and (c) shows LDA projection using absolute speaker classes.

repeat s_n as many times as there are tokens in $\mathbf{y}_n$, i.e., we want to predict a speaker label for each ASR token. Thereafter, we use the non-overlapping streams $\mathbf{H}_c$ to estimate $\hat{\mathbf{S}}_c$ in the same two-branch approach as the ASR transducer. For this, we add an auxiliary speaker transducer to each of the two branches in the recognition module, as shown in Fig. 2. Intermediate representations $\mathbf{h}_n$ from the n^{th} layer of the main encoder are fed into an auxiliary encoder, producing $\mathbf{f}^{\text{aux}}_{1:T}$. An auxiliary joiner combines $\mathbf{f}^{\text{aux}}_{1:T}$ with $\mathbf{g}_{1:U}$ to produce auxiliary logits $\mathbf{Z}^{\text{aux}} \in \mathbb{R}^{T\times U\times(K+1)}$, which are used to obtain a distribution over the speaker labels and the blank label. Combining the auxiliary encoder representation with representations from the ASR prediction network allows the speaker branch to leverage lexical content for predicting speaker labels. Such a use of lexical information has been shown to be beneficial for speaker diarization using clustering-based [33, 34] or end-to-end neural approaches [35].

3.2. Synchronizing speaker labels with ASR tokens

Since transducers perform frame-synchronous decoding with the blank label, the above formulation has several issues. First, we cannot ensure that the number of ASR tokens $|\hat{\mathbf{Y}}_c|$ predicted on branch c is equal to the the number of speaker labels $|\hat{\mathbf{S}}_c|$. Even if we can ensure this, assigning speaker labels to ASR tokens can be hard, as evident from the following example containing two speakers saying the words "hello" and "hi":

ASR	ϕ	he	ll	ϕ	o	ϕ	ϕ	hi	ϕ	ϕ
Speaker	ϕ	1	ϕ	1	ϕ	ϕ	1	2	ϕ	ϕ

Even though we predicted the correct speaker labels, it is hard to assign them to the corresponding ASR tokens since they are not synchronized by frame. To solve these problems, we need to ensure that SURT emits blank labels on the same frames for both the ASR and speaker branches. We achieve this by factoring out the blank label separately in the style of the hybrid auto-regressive transducer (HAT) model [30], i.e., we replace the alignment posterior $P(\mathbf{a}_t \mid \mathbf{f}^t_1, \mathbf{g}^{u(t)-1}_1)$ in (3) with

$$\begin{cases} b_{t,u}, & \text{if } \mathbf{a}_t = \phi, \\ (1 - b_{t,u})\ \text{Softmax}(\mathbf{z}_{t,u}[1:]), & \text{otherwise}, \end{cases} \tag{6}$$

where $b_{t,u} = \sigma(\mathbf{z}_{t,u}[0])$, and σ denotes the sigmoid function. Such factorization of the blank label has also been used for efficient language model adaptation in the factorized neural transducer model [36]. By setting $\mathbf{Z}^{\text{aux}}_{v=\phi} = \mathbf{Z}_{v=\phi}$, i.e., by sharing the blank logit for the ASR and speaker outputs, we ensure that blank emission is synchronized between the two branches. The speaker branch is trained with a similar HAT loss, i.e.,

$$\mathcal{L}_{\text{aux}} = \mathcal{L}_{\text{hat}}(\mathbf{H}_1, \mathbf{Z}^{\text{aux}}_1) + \mathcal{L}_{\text{hat}}(\mathbf{H}_2, \mathbf{Z}^{\text{aux}}_2). \tag{7}$$

Such a synchronization strategy has also recently been proposed for performing word-level diarization using transducers [37]. For both ASR and speaker branches, we use a pruned version of the HAT loss similar to pruned RNNT [31].

3.3. Maintaining state across utterance groups

A common approach for inference of long-form audio is by chunking in some way (e.g., at silences or fixed-length chunks), processing each chunk separately, and then combining the outputs. For SURT, we assume that the recording has been chunked at silences to create utterance groups, which are sets of utterances connected by speaker overlaps. For multi-talker ASR methods such as SA-ASR (discussed in Section 1) which predict *absolute* speaker identities using external speaker profiles, combining chunk-wise outputs is relatively straightforward since there is no issue of speaker label permutation. However, the auxiliary speaker branch in SURT is trained to predict *relative* speaker labels in FIFO order within that chunk, and these labels must be reconciled across all chunks within a recording in order to obtain the final speaker-attributed transcript.

3.3.1. What does the auxiliary encoder encode?

Speaker label reconciliation across different chunks for long-form diarization or speaker-attributed ASR is often done through clustering of speaker embeddings estimated from the chunks. For example, the EEND-VC model for speaker diarization extends EEND for diarization of long-recordings by applying clustering over chunk-wise speaker vectors [38]. This method delays the output prediction at least until the end of the chunk so that the re-clustering may be done. To remedy this issue, SA-ASR based on t-SOT estimates speaker change based on cosine similarity between consecutive speaker vectors, and applies re-clustering of all vectors every time a speaker change is detected [21]. Nevertheless, solving label permutation through such clustering requires that the chunk-wise speaker vectors should represent absolute speaker identities. This requirement may not be satisfied in the SURT model since the auxiliary speaker branch is trained to predict relative speaker labels in their order of appearance in the mixture.

To verify this, we applied SURT with the auxiliary speaker branch on synthetic mixtures of LibriSpeech utterances called LSMix (described in Section 4.2) consisting of 2 or 3 speakers per mixture. We collected the 256-dimensional encoder representations for the frames where the auxiliary branch predicts a speaker label, and averaged each speaker's embeddings over the mixture. In Fig. 3, we show UMAP and LDA projections of these embeddings for 15 different speakers in the LSMix `dev` set. In Fig. 3(a), the three colors denote the relative speaker label assigned to the speaker during SURT inference and the

markers denote absolute speaker identities. Fig. 3(b) shows the same plot, but in this case colors denote absolute speaker identities and markers denote relative order within the chunk. It is easy to see that the embeddings cluster by relative speaker labels instead of absolute speaker identities, validating our conjecture that the auxiliary encoder extracts relative speaker position in the chunk. Even when LDA using absolute speaker labels is used for the low-dimensional projection (as shown in Fig. 3(c)), we did not find clusters of absolute speaker labels. Interestingly, the embeddings did retain information about the speaker's gender. In the figure, the points with and without a black border denote female and male speakers, respectively, and they appear well-separated into gender-based clusters.

3.3.2. *The speaker prefixing method*

Inspired by the use of a speaker tracing buffer in the EEND model for online diarization [39], we propose a novel *speaker prefixing* strategy to solve the problem of speaker label permutation across utterance groups. The idea of speaker prefixing is to append high-confidence frames for speakers we have seen so far in the recording, before the chunk's input features, in the order of their predicted label. Formally, let $\{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$ be the input features corresponding to M utterance groups in a recording, such that $\mathbf{X}_m \in \mathbb{R}^{T_m \times F}$. For some chunk m, let $K_m \in [0, K]$ be the number of speakers seen so far in the recording. We define some function $\mathscr{S}$ which selects frames of a given speaker in the previous chunks, i.e.,

$$\mathscr{S}(\mathbf{X}_1, \ldots, \mathbf{X}_{m-1}, k) = \mathbf{B}_k, \tag{8}$$

where k is one of the K_m speakers and $\mathbf{B}_k \in \mathbb{R}^{\tau \times F}$, for some τ (which is a hyperparameter), is analogous to a speaker "buffer." Then, the speaker-prefixed input for chunk m is given as

$$\tilde{\mathbf{X}}_m = \left[\mathbf{B}_1^\top; \ldots; \mathbf{B}_{K_m}^\top; \mathbf{X}^\top\right]^\top, \tag{9}$$

where $\cdot^\top$ denotes transpose. We use $\tilde{\mathbf{X}}_m$ instead of $\mathbf{X}_m$ as input for this chunk with the conjecture that the speaker buffers would enforce a relative ordering among speakers in the current chunk. At the output of the main and auxiliary encoders, we remove the representation corresponding to the prefix, which is of length $\frac{K_m \times \tau}{s}$, where s is the subsampling factor of the encoder. During inference, we set $\mathscr{S}$ to select a sequence of τ frames (from the previous chunks) with the largest sum of confidence value, as predicted by its logit $\mathbf{z}^{\text{aux}}[k]$. During training, we randomly select κ speakers to prefix from all speakers in the batch. Such a strategy mimics the expected inference time scenario, where not all prefixed speakers will be seen in every chunk. For each selected speaker, we randomly sample a range of τ frames from all the segments of that speaker.

4. Experimental Setup

4.1. Network architecture

The main SURT model follows earlier work [23]. The masking network comprises four 256-dim DP-LSTM layers [40]. Masked features are reduced to half the original length through a convolutional layer, and the subsampled features are fed into a zipformer encoder [41]. The ASR encoder consists of 6 zipformer blocks subsampled at different frame rates (up to 8x in the middle). The encoder output is further down-sampled such that the overall subsampling factor is 4x. The representations from an intermediate layer of the ASR encoder are passed to the auxiliary encoder. This is another zipformer comprising 3 blocks with smaller attention and feed-forward dimensions. Branch tying is used at the output of both encoders using unidirectional LSTM layers [23]. The ASR prediction network contains a single 512-dim Conv1D layer. The complete SURT model contains 38.0M parameters, divided up into 6.0M, 23.6M, and 8.4M for the masking network, the ASR branch, and the speaker branch, respectively. The chunk size for the intra-LSTM and the zipformer is set to 32 frames, resulting in a modeling latency of 320 ms.

Table 1: Statistics of datasets used for evaluations. The overlap durations are in terms of fraction of total speaking time.

	LSMix		AMI		
	Train	**Dev**	**Train**	**Dev**	**Test**
Duration (h:m)	2193:57	4:19	79:23	9:40	9:03
Num. sessions	486440	897	133	18	16
Silence (%)	3.4	3.2	18.1	21.5	19.6
Overlap (%)	28.4	26.0	24.5	25.7	27.0

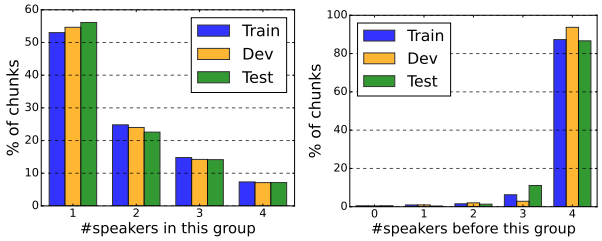


Figure 4: Utterance group statistics of the AMI meeting corpus: (a) number of speakers in the group, and (b) number of speakers seen before the group. An utterance group is defined as a set of overlapping segments, and different utterance groups are separated by a pause.

4.2. Data

We conducted our experiments on synthetically mixed LibriSpeech utterances (called LSMix) and the AMI meeting corpus, and their statistics are shown in Table 1. To create LSMix, we first cut LibriSpeech utterances at 0.2 second pauses, and then mixed speed-perturbed versions of these segments using the algorithm described in [23]. The resulting mixtures were 17s long on average, and contain 2–3 speakers and up to 9 turns of conversation. We created `train` and `dev` splits of LSMix using the corresponding LibriSpeech partitions. We used this evaluation data to perform ablations for developing the auxiliary speaker branch in SURT.

AMI consists of 100 hours of recorded meetings containing 4 or 5 speakers per session [4]. Sessions were recorded on close-talk (headset and lapel) microphones, as well as 2 linear arrays each containing 8 mics. We used three different mic settings for our experiments: IHM-Mix (digitally mixed individual headset mics), SDM (first channel of array-1), and MDM (beamformed array-1), where the last setting uses officially provided beamformed recordings [42]. To train SURT models for AMI, we used synthetic mixtures of AMI and ICSI [43] utterances (known as AIMix) as described in [23]. We first trained the models on 1841h of the AIMix data, and then adapted them on real `train` sessions. For training with the speaker buffer, we fixed τ as 128 frames for each speaker, and chose K_m from [0, 4] with probabilities [0.05, 0.05, 0.1, 0.2, 0.6]. This is because during inference, most chunks will be processed with 4 speaker buffers, as seen in Fig. 4.

4.3. Training details

The auxiliary loss scales in (5), λ_{ctc} and λ_{mask}, were set to 0.2 each. We trained the models with the ScaledAdam optimizer following the standard zipformer-transducer recipes in

Table 2: Comparison of different training strategies for SURT with auxiliary speaker branch.

Strategy	ORC-WER	WDER	cpWER
Sequential	8.53	**3.99**	14.96
Joint	**8.43**	4.46	**14.95**
Seq. + Joint	9.17	4.25	15.33

`icefall` [41]. This is a variant of Adam where each parameter's update is scaled proportional to the norm of that parameter. The learning rate was warmed up to 0.004 for 5000 iterations, and decayed exponentially thereafter. As described in Section 5.2, we tried sequential and joint training strategies. For the former, the SURT model was trained for 40 epochs. For the latter, the ASR branch was first trained for 30 epochs; it was then frozen and the speaker branch was trained for 20 epochs. In all cases, the ASR transducer was initialized from a pre-trained transducer model, trained for 10 epochs on LibriSpeech, since this has been found useful for fast convergence [23]. We averaged model checkpoints from the last 5 epochs for inference, and used greedy decoding for reporting all results. For evaluation on AMI, we initialized the masking network and the ASR branch using the parameters from the SURT model trained on LSMix. We then trained this model in a sequential process, i.e., ASR branch followed by speaker branch, on AIMix followed by adaptation on real AMI training sessions.

4.4. Evaluation

For speaker-agnostic transcription, SURT was evaluated using the optimal reference combination word error rate (ORC-WER) metric, proposed independently in [25] and [26]. ORC-WER computes the minimum total WER obtained using the optimal assignment of reference utterances to the output channels. In this paper, since we have extended SURT to perform speaker-attributed transcription, we measure its performance using the concatenated minimum-permutation WER (cpWER) [3]. This metric finds the best permutation of reference and hypothesis speakers which minimizes the total WER across all speakers.

We also want to measure speaker attribution errors independently of transcription errors. The conventional metric for this is known as diarization error rate (DER), and measures the duration ratio of speaking time for which the predicted speakers do not match the reference speakers. However, since SURT is a streaming model, the ASR tokens and the respective speaker labels may be emitted with some latency compared to their actual reference time-stamp. This can artificially escalate the DER even when there are few speaker attribution errors. To circumvent this issue, we report a word-level diarization error rate (WDER) inspired by [44]. Originally, WDER was defined as the fraction of correctly recognized words which have incorrect speaker tags. We modify the metric for SURT by using the ORC-WER reference assignment to identify the correct words and the speaker mapping from the cpWER computation to check for speaker equivalence.[1]

5. Results & Discussion

5.1. RNN-T vs. HAT for speaker-agnostic ASR

Since our formulation requires replacing the conventional RNN-T loss, i.e. (3), with the HAT loss given by (6), we want to ensure that the speaker-agnostic ASR performance of the model does not degrade. To verify this, we trained SURT (without an auxiliary branch) using $\mathcal{L}_{\text{rnnt}}$ and $\mathcal{L}_{\text{hat}}$ on the LSMix `train` set, and evaluated the resulting models on the `dev` set. We found that the **HAT model obtained 8.53% ORC-WER**, compared to 8.59% using regular RNN-T. The error breakdown showed marginally higher insertions but fewer deletions, which may be due to explicit modeling of the blank token.

Table 3: Speaker-attributed ASR performance on LSMix `dev` for different positions of the auxiliary encoder. $\mathbf{h}_l$ denotes the hidden representation at the l^{th} block of the main zipformer encoder, and $\mathbf{h}_0^{\text{aux}}$ is the input to the auxiliary encoder.

$\mathbf{h}_0^{\text{aux}}$	Ins.	Del.	Sub.	cpWER	WDER
$= \mathbf{h}_0$	3.34	6.04	7.28	16.66	5.36
$= \mathbf{h}_1$	**2.91**	**5.20**	**6.85**	**14.96**	**3.99**
$= \mathbf{h}_2$	4.58	6.77	8.24	19.59	6.73
$= \mathbf{h}_3$	5.95	8.23	9.41	23.59	8.35

5.2. Sequential vs. joint training

The auxiliary speaker branch of the SURT model can be trained in several ways, as shown in Table 2. In "sequential" training, the main SURT model is first trained using (5) and then frozen while the auxiliary branch is trained using (7). In "joint" training, the full model is trained from scratch with the multi-task objective. Finally, we can combine the above approaches by first training the branches sequentially and then fine-tuning them jointly. We found that **both sequential and joint training resulted in similar cpWER performance**, but joint training degrades WDER. Furthermore, joint fine-tuning after sequential training degraded performance on both ASR metrics. Since sequential training allows decoupling of ASR and speaker attribution performance, we used this strategy for the experiments in the remainder of this paper.

5.3. Auxiliary encoder position

The input $\mathbf{h}_0^{\text{aux}}$ to the auxiliary encoder is obtained from an intermediate representation of the main encoder. We trained several SURT models with different positions for the auxiliary input, in order to find the optimal representation for the speaker branch, and the results are shown in Table 3. The models were trained sequentially and the ORC-WER was 8.53% (same as earlier). We found that both cpWER and WDER get progressively worse if we used representations from deeper layers, possibly because of loss in speaker information through the main encoder. Interestingly, $\mathbf{h}_1$ (i.e. output of the first zipformer block) showed better performance than $\mathbf{h}_0$ (output from convolutional embedding layer). We conjecture that the input to the **auxiliary encoder needs contextualized representations** since speaker labels need to be synchronized across the two branches. These findings mirror recent studies showing that intermediate layers of the acoustic model are most suitable for extracting speaker information [37]. Such analysis has also motivated "tandem" multi-task learning of ASR and speaker diarization using self-supervised encoders such as Wav2Vec 2.0 [45].

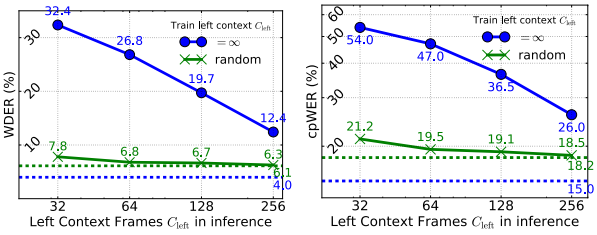
5.4. Effect of left context

The ASR encoder of the SURT model uses limited left context (C_{left}=128 frames) in the self-attention computation during inference. While ASR token prediction is usually a local decision, speaker label prediction requires looking at the full history in order to synchronize the relative FIFO labels. We experimented with training and decoding with different histories, and the results are shown in Fig. 5. For a model trained with infinite C_{left}

[1] We use ORC-WER instead of cpWER to find word alignment to be as close as possible to the original WDER metric, which first obtains an alignment without the knowledge of speaker labels, and thereafter computes the speaker confusion error.

Table 4: Performance of SURT models (with and without speaker prefixing) for different conditions on AMI `test` set, evaluated on utterance groups. "ORC" denotes the ORC-WER metric, and is the same for all models.

	Prefix	IHM-Mix			SDM			MDM		
ID	**Train/Decode**	ORC	WDER	cpWER	ORC	WDER	cpWER	ORC	WDER	cpWER
A	✗ / ✗	34.9	**9.3**	**42.9**	43.2	**10.9**	**50.3**	40.5	**9.9**	**47.3**
B	✗ / ✓	34.9	22.5	61.2	43.2	23.1	68.2	40.5	22.6	64.8
C	✓ / ✓	34.9	14.0	49.9	43.2	16.3	58.9	40.5	15.5	56.0



(a) WDER **(b)** cpWER

Figure 5: Effect of auxiliary encoder left context on (a) WDER and (b) cpWER. Dotted lines show best performance using ∞ left context.

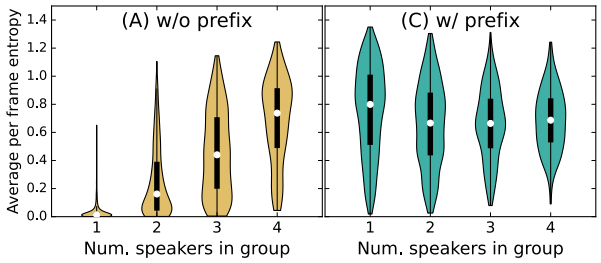


Figure 6: Average per-frame entropy for utterance groups with different number of speakers.

(solid blue line), limiting it during inference quickly degraded WDER and cpWER performance. When the model was trained with randomized C_{left} sampled uniformly from $\{64, 96, 128\}$ (solid green line), the degradation was less evident. However, it was unable to make full use of infinite history at inference time, and only obtained a WDER of 6.12%, versus 3.99% for the model trained with infinite C_{left}. This indicates that using **infinite left context during training and inference is important for the auxiliary speaker encoder**.

5.5. Utterance-group evaluation on AMI

We evaluated the SURT model on different microphone settings of the AMI meeting corpus in the utterance-group scenario, and the results are shown in Table 4 in terms of ORC-WER, WDER, and cpWER. Across the board, performance degraded from IHM-Mix to SDM settings, which is expected since SDM contains far-field artifacts in addition to overlaps. Beamforming with multiple microphones partially removes background noise and reverberations, thus providing a slightly easier condition than SDM. For system A, which was trained and decoded without speaker prefixing, we obtained a cpWER of 46.8%, on average across the three conditions. When we used the same model for decoding with speaker prefixes (system B), the cpWER performance degraded by 38.2% relative to the former. Since the model has not seen short speaker buffers at train time, the auxiliary encoder is not adept at using these for generating the contextualized representations.

Next, we trained the same SURT model using speaker prefixing as described in Sections 3.3.2 and 4.2, and found that it

Table 5: Breakdown of model (A)'s performance on IHM-Mix `test` set by number of speakers in the utterance group.

#spk	**1**	**2**	**3**	**4**	**Avg.**
WDER (↓)	0.1	3.4	13.0	23.9	9.3
Count. (↑)	98.6	61.9	26.8	44.0	75.9
cpWER (↓)	17.2	32.4	51.1	63.6	42.9

Table 6: Full-session evaluation cpWER (%) on AMI `test` set.

Method	**IHM-Mix**	**SDM**	**MDM**
w/o prefix	100.11	97.15	96.26
w/ prefix	82.77	83.94	82.28
+ enrollment	53.04	61.22	59.59

improved performance significantly due to matched train and test conditions. Nevertheless, this model was **7-8% worse than the original** model in terms of absolute cpWER performance. To investigate this further, we computed the average framewise entropy over speaker labels for all utterance groups in the IHM-Mix `test` set, and grouped them by number of speakers in the group. Fig. 6 shows the distribution of these entropies for the SURT model with and without speaker prefixing. We found that for the model without prefixing, the **entropy was very low for utterance groups with a single speaker**, and gradually increased with the number of speakers. This indicates that the model was very confident in its prediction for few speaker cases. The opposite trend was seen for the model with speaker prefixing, where the entropy was highest for the single-speaker case. This is because for each frame, the model needs to decide which of the 4 prefixed speakers the frame should be assigned to, which may result in low confidence of prediction.

In general, we found that the performance of all models **gets progressively worse** as the number of speakers in the group increases, as shown in Table 5 for system A. Interestingly, the degradation in WDER was small compared to that in the speaker counting accuracy. This may be because of several utterance groups where some speakers participate with just a few words, which may be hard for the system to identify, but do not contribute much in overall speaker attribution error.

5.6. Full-session evaluation on AMI

Finally, we performed inference on full AMI `test` sessions and the corresponding cpWERs are reported n Table 6. Computing the ORC-WER and WDER for this case was not feasible since their computational complexity depends on the number of segments in the reference. First, we see that the model without speaker prefixing obtained very high error rates, since it failed at correctly reconciling speaker labels across different utterance groups. With speaker prefixing, we obtained **15.1% relative cpWER improvement** on average across the mic settings. For the speaker prefixing, we trained and evaluated the model using τ of 128 frames or 1.28s per speaker.

In a meeting transcription setup, since the participants are known before-hand, we can usually obtain an enrollment utterance for each speaker. Instead of selecting speaker prefixes from previous chunks, if we select them from these enrollment utterances, we obtain a further **relative cpWER improvement of 29.2%**, on average. We conjecture that when enrollment utterances are not used, speaker attribution errors in earlier chunks can adversely impact performance on the current chunk, since the buffer frames are used to guide the relative order. Nev-

ertheless, there still exists a significant gap of about 10–12% absolute cpWER between full session evaluation and utterance group evaluation (shown in Table 4).

6. Conclusion

The SURT framework allows continuous, streaming recognition of multi-talker conversations, but it could only be used for speaker-agnostic transcription. In this paper, we showed how to perform streaming word-level speaker labeling with SURT, thus enabling speaker-attributed transcription using the same model. We achieved this by adding an auxiliary speaker encoder to the recognition component of the model, and used the same two-branch strategy to handle overlapping speech. We solved the problem of synchronization between the ASR and speaker branch outputs by factoring out the blank logit and sharing it between the branches. Since the model predicts relative speaker labels in FIFO order, reconciling the labels across utterance groups in a recording becomes a challenge. We showed that a simple strategy of prefixing high-confidence speaker frames for the recognized speakers can partially alleviate this problem, but it would require further investigation to bring session-level error rates closer to those for utterance groups.

7. References

[1] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines," in *IEEE ASRU*, 2015.

[2] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, A. Sehr, W. Kellermann, and R. Maas, "The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech," in *IEEE WASPAA*, 2013.

[3] S. Watanabe, M. Mandel, J. Barker, and E. Vincent, "CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings," *ArXiv*, 2020.

[4] J. Carletta et al., "The AMI meeting corpus: A pre-announcement," in *MLMI*, 2005.

[5] E. Shriberg, A. Stolcke, and D. Baron, "Observations on overlap: findings and implications for automatic processing of multi-party conversation," in *Interspeech*, 2001.

[6] T. Yoshioka, D. Dimitriadis, A. Stolcke, W. Hinthorn, Z. Chen, M. Zeng, and X. Huang, "Meeting transcription using asynchronous distant microphones," in *Interspeech*, 2019.

[7] D. Raj, D. Povey, and S. Khudanpur, "GPU-accelerated guided source separation for meeting transcription," in *Interspeech*, 2023.

[8] D. Raj, P. Denisov, et al., "Integration of speech separation, diarization, and recognition for multi-speaker meetings: System description, comparison, and analysis," in *IEEE SLT*, 2021.

[9] N. Kanda, S. Horiguchi, Y. Fujita, Y. Xue, K. Nagamatsu, and S. Watanabe, "Simultaneous speech recognition and speaker diarization for monaural dialogue recordings with target-speaker acoustic models," in *IEEE ASRU*, 2019, pp. 31–38.

[10] J. Wu, Z. Chen, S. Chen, Y. Wu, T. Yoshioka, N. Kanda, S. Liu, and J. Li, "Investigation of practical aspects of single channel speech separation for ASR," in *Interspeech*, 2021.

[11] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[12] N. Kanda, Y. Gaur, X. Wang, Z. Meng, and T. Yoshioka, "Serialized output training for end-to-end overlapped speech recognition," in *Interspeech*, 2020.

[13] N. Kanda, Y. Gaur, et al., "Joint speaker counting, speech recognition, and speaker identification for overlapped speech of any number of speakers," in *Interspeech*, 2020.

[14] N. Kanda, G. Ye, Y. Gaur, X. Wang, Z. Meng, Z. Chen, and T. Yoshioka, "End-to-end speaker-attributed ASR with transformer," in *Interspeech*, 2021.

[15] N. Kanda, X. Xiao, J. Wu, T. Zhou, Y. Gaur, X. Wang, Z. Meng, Z. Chen, and T. Yoshioka, "A comparative study of modular and joint approaches for speaker-attributed asr on monaural long-form audio," in *IEEE ASRU*, 2021, pp. 296–303.

[16] X. Chang, N. Kanda, Y. Gaur, X. Wang, Z. Meng, and T. Yoshioka, "Hypothesis stitcher for end-to-end speaker-attributed ASR on long-form multi-talker recordings," in *IEEE ICASSP*, 2021.

[17] N. Kanda, X. Chang, Y. Gaur, X. Wang, Z. Meng, Z. Chen, and T. Yoshioka, "Investigation of end-to-end speaker-attributed ASR for continuous multi-talker recordings," in *IEEE SLT*, 2020.

[18] F. Yu, Z. Du, S. Zhang, Y. Lin, and L. Xie, "A comparative study on speaker-attributed automatic speech recognition in multi-party meetings," in *Interspeech*, 2022.

[19] M. Shi, J. Zhang, Z. Du, F. Yu, S. Zhang, and L. Dai, "A comparative study on multichannel speaker-attributed automatic speech recognition in multi-party meetings," in *APSIPA ASC*, 2022, pp. 1943–1948.

[20] M. Shi, Z. Du, Q. Chen, F. Yu, Y. Li, S. Zhang, J. Zhang, and L. Dai, "Casa-asr: Context-aware speaker-attributed asr," in *Interspeech*, 2023.

[21] 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 *Interspeech*, 2022.

[22] A. Graves, "Sequence transduction with recurrent neural networks," in *ICML Representation Learning Workshop*, 2012.

[23] D. Raj, D. Povey, and S. Khudanpur, "SURT 2.0: Advances in transducer-based multi-talker speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 3800–3813, 2023.

[24] L. Lu, N. Kanda, J. Li, and Y. Gong, "Streaming end-to-end multi-talker speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 803–807, 2020.

[25] I. Sklyar, A. Piunova, X. Zheng, and Y. Liu, "Multi-turn RNN-T for streaming recognition of multi-party speech," in *IEEE ICASSP*, 2021.

[26] D. Raj, L. Lu, Z. Chen, Y. Gaur, and J. Li, "Continuous streaming multi-talker ASR with dual-path transducers," in *IEEE ICASSP*, 2022.

[27] L. Lu, J. Li, and Y. Gong, "Endpoint detection for streaming end-to-end multi-talker ASR," in *IEEE ICASSP*, 2022.

[28] I. Sklyar, A. Piunova, and C. Osendorfer, "Separator-transducer-segmenter: Streaming recognition and segmentation of multi-party speech," in *Interspeech*, 2022.

[29] L. Lu, N. Kanda, J. Li, and Y. Gong, "Streaming multi-talker speech recognition with joint speaker identification," in *Interspeech*, 2021.

[30] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (hat)," in *IEEE ICASSP*, 2020, pp. 6139–6143.

[31] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and

D. Povey, "Pruned RNN-T for fast, memory-efficient ASR training," in *Interspeech*, 2022.

[32] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[33] N. Flemotomos, P. G. Georgiou, and S. S. Narayanan, "Language aided speaker diarization using speaker role information," in *Speaker Odyssey*, 2020.

[34] T. J. Park, K. J. Han, J. Huang, X. He, B. Zhou, P. G. Georgiou, and S. S. Narayanan, "Speaker diarization with lexical information," in *Interspeech*, 2019.

[35] A. Khare, E. Han, Y. Yang, and A. Stolcke, "Asr-aware end-to-end neural diarization," in *IEEE ICASSP*, 2022, pp. 8092–8096.

[36] X. Chen, Z. Meng, S. Parthasarathy, and J. Li, "Factorized neural transducer for efficient language model adaptation," *IEEE ICASSP*, pp. 8132–8136, 2022.

[37] 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*, vol. abs/2309.08489, 2023.

[38] K. Kinoshita, M. Delcroix, and N. Tawara, "Integrating end-to-end neural and clustering-based diarization: Getting the best of both worlds," in *IEEE ICASSP*, 2020, pp. 7198–7202.

[39] Y. Xue, S. Horiguchi, Y. Fujita, S. Watanabe, and K. Nagamatsu, "Online end-to-end neural diarization with speaker-tracing buffer," in *IEEE SLT*, 2021, pp. 841–848.

[40] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *IEEE ICASSP*, 2019.

[41] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, and D. Povey, "Zipformer: A faster and better encoder for automatic speech recognition," *ArXiv*, vol. abs/2310.11230, 2023.

[42] X. A. Miró, C. Wooters, and J. Hernando, "Acoustic beamforming for speaker diarization of meetings," *IEEE TASLP*, vol. 15, pp. 2011–2022, 2007.

[43] A. L. Janin, D. Baron, J. Edwards, D. P. W. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters, "The ICSI meeting corpus," in *IEEE ICASSP*, 2003.

[44] L. E. Shafey, H. Soltau, and I. Shafran, "Joint speech recognition and speaker diarization via sequence transduction," in *Interspeech*, 2019.

[45] X. Zheng, C. Zhang, and P. C. Woodland, "Tandem multitask training of speaker diarisation and speech recognition for meeting transcription," in *Interspeech*, 2022.