ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10446130

MULTITASK SPEECH RECOGNITION AND SPEAKER CHANGE DETECTION FOR UNKNOWN NUMBER OF SPEAKERS

Shashi Kumar[1], *Srikanth Madikeri*[1], *Iuliia Nigmatulina*[1,3], *Esaú Villatoro-Tello*[1], *Petr Motlicek*[1,2], *Karthik Pandia*[4], *S. Pavankumar Dubagunta*[4], *Aravind Ganapathiraju*[4]

[1]Idiap Research Institute, Martigny, Switzerland
[2]Brno University of Technology, Brno, Czech Republic
[3] Institute of Computational Linguistics, University of Zürich, Switzerland
[4] Uniphore Software Systems Inc., Palo Alto, CA, USA.

ABSTRACT

Traditionally, automatic speech recognition (ASR) and speaker change detection (SCD) systems have been independently trained to generate comprehensive transcripts accompanied by speaker turns. Recently, joint training of ASR and SCD systems, by inserting speaker turn tokens in the ASR training text, has been shown to be successful. In this work, we present a multitask alternative to the joint training approach. Results obtained on the mix-headset audios of AMI corpus show that the proposed multitask training yields an absolute improvement of 1.8% in coverage and purity based F1 score on SCD task without ASR degradation. We also examine the trade-offs between the ASR and SCD performance when trained using multitask criteria. Additionally, we validate the speaker change information in the embedding spaces obtained after different transformer layers of a self-supervised pre-trained model, such as XLSR-53, by integrating an SCD classifier at the output of specific transformer layers. Results reveal that the use of different embedding spaces from XLSR-53 model for multitask ASR and SCD is advantageous.[1]

***Index Terms*—** speaker change detection, speaker turn detection, speech recognition, multitask learning, F1 score

1. INTRODUCTION

Speaker change detection (SCD) is the task of finding precise time instances in audio where speaker transitions occur. The SCD systems find utility in many tasks such as speaker diarization (SD) [1, 2], measuring speaking time per speaker, and speaker-specific emotion analysis for tracking customer-agent interactions [3] in call center analytics. In many of these use cases, a precise speaker change timestamp is required.

Conventionally, separate systems for automatic speech recognition (ASR) and SCD tasks [1, 4, 5, 6, 7] have been explored. In many practical scenarios, it may be infeasible to train and maintain independent models for ASR and SCD. Recently, joint training of ASR and SCD [8, 9, 10, 11, 12] has been proposed. The joint system trains ASR by augmenting reference text with speaker turn tokens. There are various limitations to this approach. First, the tight coupling of SCD with ASR was observed to result in error propagation. Second, the SCD model can not be tuned independently to adapt to various speech variability unseen during training. Third, popular end-to-end (E2E) ASR training frameworks are known to output incorrect word timestamps [13, 14], leading to erroneous speaker turn estimation. Fourth, usually no specific focus is given to speaker turn token during ASR training; despite some recent works [10, 11] in this direction, they are still limited by ASR model capacity.

In this paper, we propose multitask training of ASR and SCD where each task is learnt by a separate head taking input from a shared model. The multitask architecture enables task specific scaling and fine-tuning while mitigating the effect of ASR on the SCD performance at test time. Moreover, we train the SCD model at the acoustic frame level which typically has lower time granularity and does not rely on the ASR output. The multitask model is trained using a weighted linear combination of the task specific losses where weights can be tuned for individual tasks. Inspired by the success of self-supervised pre-trained models in speech tasks [15, 16], we use XLSR-53 [15] as our base model. Speaker variability may hurt the ASR performance [17]. It may be beneficial to use different embedding spaces from XLSR-53 model as input to ASR and SCD models. We study this by attaching the SCD model at the output of different transformer layers in XLSR-53 model while keeping the ASR model fixed at the output of the last layer. Finally, we propose to combine both the joint training approaches to train a multitask model with speaker turn token in train text data. Overall, we show that the proposed multitask training improves the SCD without noticeable degradation in the ASR.

2. JOINT ASR AND SCD

The section delves into a robust baseline for the joint system, followed by the introduction of our multitask approach.

2.1. ASR with speaker token

In [8], the authors propose introducing speaker-specific tokens into the training text. In this paper, our focus is on de-

*Corresponding author: *shashi.kumar@idiap.ch*

This work was supported by the Idiap & Uniphore collaboration project.

[1]We release scripts for data processing and model evaluation: `https://github.com/idiap/multitask_asr_and_scd`

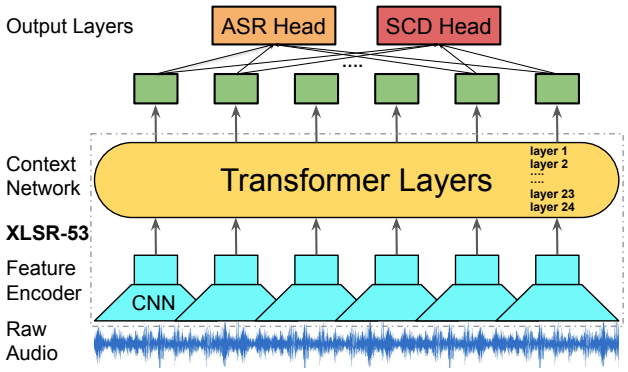

Fig. 1. Multitask ASR and SCD model using pre-trained XLSR-53 as base.

tecting speaker turns which allows us to use a single speaker turn token [11] <spk> to be inserted into reference text for ASR training. During inference, the predicted timestamps for the word <spk> in the hypothesis are used as the speaker change points in the audio. This system acts as a baseline for the proposed method.

2.2. Proposed multitask approach

As shown in Figure 1, an SCD Head is attached to the output of the XLSR-53 model. It takes audio embedding from the XLSR-53 model as input and predicts the probability of speaker change for each frame. This multitask model is trained with an ASR objective applied at the output of the ASR head and a two-class classification loss at the output of the SCD head. For all our experiments, we use E2E-LFMMI [18] loss to train our ASR system and binary cross entropy (BCE) loss to train the SCD classifier. The final loss which is used to train the multitask model is given by $\mathbf{L} = \mathbf{E2E\text{-}LFMMI} + \lambda * \mathbf{BCE}$, where λ controls the effect of the SCD classifier on overall training. In BCE loss calculation, class 1 correspond to frames with speaker changes. It is reasonable to assume that speaker changes are much lower than number of embedding frames in an audio, this could lead to ineffective SCD classifier training due to a severe class imbalance. To handle this imbalance, following [19] we label ± 10 frames as class 1 for each speaker change timestamp.

During inference, output from both the ASR and SCD heads is obtained. The output from the ASR head is processed to arrive at the hypothesis text. A threshold is used on the output from SCD head to decide whether a frame belongs to class 1. The consecutive frames of class 1 are defined as a chunk and we extract all such chunks for an utterance. Following the labeling paradigm during training, if the length of a chunk is less than 21 frames, the timestamp corresponding to the middle frame is the predicted speaker change timestamp with a duration equal to half of the chunk duration. But if the length of a chunk is more than 21 frames, we simultaneously loop from start and end in steps of 21 frames till these two directions meet or overlap and similarly calculate the turn timestamps and duration for all such splits.

3. DATASET

AMI [20] dataset has been extensively used in past for analysing meetings specifically for ASR and keyword spotting [21], including new approaches to remove the mismatch between close-talking and distant-talking conditions [22]. Kaldi [23] s5b recipe for AMI [20] is used across this paper, which includes audios sampled at 16 kHz. It contains train, dev, and eval splits for individual head microphone (IHM), termed as IHM$_{orig}$. The default segments file contains utterances from one speaker which can not be used for SCD. Hence, we prepare a synthetic dataset containing varied number of speakers in a single utterance from default splits.

Synthetic Data (IHM$_{syn}$): We begin by sorting all the segments by the start time for each meeting. For each segment in a meeting, we read raw audio between the segment endpoints from the corresponding microphone. Reference texts for each segment are taken from the IHM$_{orig}$. The audio and text segments are concatenated until the combined audio duration is within the maximum limit. We choose 20s as the maximum duration of an utterance due to the relatively high memory requirement of the XLSR-53 model during training. If there is a speaker change, the concatenation point in audio serves as the change timestamp and the <spk> is inserted into reference texts. The concatenated audio and text are then dumped to disk as a single utterance. This process is performed for all splits in the IHM$_{orig}$ dataset. Note that the precise speaker change point is known for all utterances in the IHM$_{syn}$ dataset which is required to train our multitask model. Also, no randomness is introduced and no segment is rejected. In the IHM$_{syn}$ dataset, approximately 7.2% of words in the train set, 7.5% in the dev set, and 7.6% in the eval set are <spk> tokens. The number of speaker changes in an utterance ranges from 0 to 18, with 84% in the train set, 83% in the dev set, and 81% in the eval set containing at least one speaker change. The wide range of speaker turns per utterance provides diverse scenarios for training and evaluation.

AMI Mix-Headset (IHM$_{mix}$): To compare the performance of the baseline and multitask model on real conversation data, we evaluate on the "mix-headset" audios from the AMI corpus. The audios are segmented into chunks of maximum 20 s while respecting the original segment boundaries so as to avoid cutting in between a segment. We concatenate the text of segments within a chunk according to their start time and insert <spk> token if there is a speaker change.

4. EVALUATION METRICS

We use the standard word error rate (WER) to measure the ASR performance. For SCD, precise speaker change timestamps are needed for many use cases. We measure timestamp precision (P), recall (R), and F1 score to reflect the same. For IHM$_{mix}$ test sets, measuring timestamps precision and recall is ambiguous due to overlap. So we report coverage (Cov), purity (Pur), and their F1 score [4] for all our experiments. These metrics are collectively termed SCD metrics.

We use a simple version of the Hungarian Matching algorithm [24] to calculate precision and recall owing to the monotonous nature of the token timestamps. The speaker change timestamps are first sorted by time for both reference and hypothesis and then matched between the two. A collar of 250 ms is used during timestamp matching, which is a common practice in speaker diarization literature [2]. If a timestamp from reference is matched with the hypothesis, it is considered true positive (TP) else false negative (FN). Any timestamp in the hypothesis not matched with reference is counted as a false positive (FP). Precision and recall are calculated using their standard definition.

Segment coverage, purity and their F1 score [4] have been commonly used evaluation metrics for SCD. Compared to the reference, detecting an excessive number of speaker changes would yield high purity but low coverage, whereas missing many speaker changes would reduce purity. The harmonic mean of these metrics is referred as the F1 score, which serves as a good indicator for assessing the accuracy of both the number of speaker changes and the corresponding timestamp predictions within the hypothesis. We use `pyannote.metrics` [25] to compute these metrics.

5. EXPERIMENTAL SETUP

To validate our experimental setup, we begin by training separate ASR and SCD systems. The ASR system is trained on IHM$_{orig}$ dataset and the SCD system is trained on the IHM$_{syn}$ dataset. We also train a standalone ASR system on the IHM$_{syn}$ without speaker token in the reference text which may serve as a sanity test of the synthetic dataset. We utilize PyTorch [26] and Fairseq [27] toolkits to train all our models.

ASR system: Following [28], our acoustic model (AM) uses the pre-trained XLSR-53 model as base, followed by ASR head. The ASR head consists of 3 factorized time-delay neural network (TDNN-F) [29] layers. We use PkWrap [30] toolkit for TDNN-F layer. The AM model is trained on biphone units using the E2E-LFMMI criteria. We use Adam [31] optimizer with a weight decay of 1e-2 and train for $35'000$ steps. Initially, the learning rate is set to 3e-5, then increased linearly for the first 10% of the steps, held constant for the next 40%, and decreased linearly for the last 50% of the steps. We use the Espresso [32] toolkit for E2E-LFMMI loss calculation which relies on PyChain [33] to implement the LFMMI loss. A 3-gram language model (LM) model is trained by following the Kaldi s5b recipe for the AMI dataset and IHM mic type. For decoding, the WFST decoder from Kaldi is used with a beam width of 15. Throughout this paper, the same model architecture, training, and decoding setup is used to train and decode the ASR system.

SCD system: A standalone SCD model employs an SCD head, comprising one input layer and one output layer, on the pre-trained XLSR-53 model. The input layer consists of a linear layer (output size: 512) followed by the GELU [37] activation function and LayerNorm [38] The output layer includes a linear layer and sigmoid activation, producing speaker change probabilities for each frame. The SCD model is trained using BCE loss. We use the same architecture for the SCD head throughout the paper. Both during training and inference, we apply the labeling and post-processing techniques described in Section 2.2.

Table 1. ASR comparison, in WER (%), on the standard dev and eval sets of AMI IHM with the AMI recipes available from popular toolkits. Our ASR model consists of TDNN-F layers stacked after the pre-trained XLSR-53 model and trained using the E2E-LFMMI objective.

Recipe	dev	eval
Kaldi s5b [34]	18.9	19.3
Espnet egs2 [35]	17.7	16.5
K2-Icefall [36]	18.9	17.4
XLSR-TDNNF-LFMMI (ours)	**14.6**	**12.6**

5.1. ASR with speaker token (baseline)

We use the same model architecture and training setup as the ASR system but the AM is trained on IHM$_{syn}$ dataset. It may be noted here that the speaker tag <spk> is not present in our vocabulary. We add a new phone <spk> as a non-silence phone in our phone set, add this token in the lexicon as a word pronounced by the new phone, and re-train our Kaldi style lang. This lang is used to build the numerator and denominator graphs required by the E2E-LFMMI objective. We train a 3-gram LM using text from IHM$_{syn}$ train split which is used for WFST-based decoding for all our experiments when the speaker token is present in text. We generate the ctm files during decoding and use the timestamps of the predicted <spk> tokens in hypothesis as speaker change points.

5.2. Multitask ASR and SCD

We train the proposed multitask model on the IHM$_{syn}$ dataset after removing <spk> tags from reference text and use the same training setup as the ASR system. At inference time, the output of the ASR head is used for WFST decoding to arrive at hypothesis text and the output of the SCD head is further processed to calculate speaker change metrics. We train a 3-gram LM on IHM$_{syn}$ train data without speaker tokens, using it for decoding for all experiments where model is trained on this data but speaker tokens are absent in the text. Same training and decoding setup is used in all multitask experiments. Further, we combine both the joint training approaches and train a multitask model on the IHM$_{syn}$ dataset with <spk> present in the reference text. The lang and LM described in Section 5.1 are used for training and decoding. Note that the SCD metrics can be calculated from the predicted speaker token in the hypothesis or from the SCD head output.

6. RESULTS

We begin by benchmarking our ASR system against publicly available results from popular toolkits on the AMI dataset.

Table 2. Comparison of ASR and SCD tasks on metrics described in Section 4 for baseline and proposed systems. All models are trained on the synthetic data except the ASR† model which is trained and tested on IHM$_{orig}$. Here, P is precision, R is recall, Cov is coverage and Pur is purity. All numbers are in percentage (%).

System	<spk> token in text	WER	P	R	F1 (P&R)	Cov	Pur	F1 (Cov &Pur)
Separate Models, tested on synthetic eval set								
ASR†	✗	**12.6**	-	-	-	-	-	-
ASR	✗	13.0	-	-	-	-	-	-
SCD	✗	-	96.3	97.0	96.6	**99.0**	**98.9**	**98.9**
Joint ASR and SCD, tested on synthetic eval set								
ASR	✓	**13.1**	92.4	94.9	93.6	96.3	97.3	96.6
multitask	✗	13.8	93.8	**97.2**	95.5	98.5	**98.9**	**98.7**
multitask	✓	**13.2**	-	-	-	-	-	-
– <spk>	-	-	**98.0**	95.6	**96.8**	96.4	96.5	96.4
– SCD head	-	-	94.8	96.7	95.8	**98.7**	**98.9**	**98.7**
Joint ASR and SCD, tested on IHM Mix-Headset eval set								
ASR	✗	**26.0**	-	-	-	-	-	-
ASR	✓	29.9	-	-	-	**95.9**	79.6	85.8
multitask	✗	**26.3**	-	-	-	92.1	**85.3**	**87.6**

From Table 1, it can be seen that our ASR system *XLSR-TDNNF-LFMMI* shows significantly better ASR results. A possible reason is that unlike other recipes which start with supervised training from scratch, we exploit a pre-trained XLSR-53 model which shows competitive results [15] even when trained with a very little amount of supervised data. In Table 2, results for baselines and proposed methods are reported. Due to space constraints, here we only report the results on the eval set. It can be seen that the joint training slightly degrades the ASR performance as previously reported in [8]. At the same time, the proposed multitask approaches improve the speaker change detection metrics significantly. Compared to the baseline system when ASR is trained on text augmented with speaker-turn token, on synthetic data, we observe an absolute improvement of 1.9% in F1 score (P&R) and 2.1% in F1 score (Pur&Cov). On IHM$_{mix}$ set, precision and recall calculation is ambiguous due to overlap where precise speaker change timestamp is not available, so we only measure coverage and purity. We observe an absolute improvement of 1.8% in their F1 score. Further, we train the multitask model with <spk> token in the reference text. Achieved results suggest that multitask training improves the performance of SCD even when the metrics are calculated from the speaker token predicted in the hypothesis. Interestingly, we observe high precision using the speaker tokens and high recall when using the output of the SCD head which may be exploited further for overall prediction.

Table 3. ASR and SCD tasks with given metrics for multitask model trained without <spk> in text, with input to the SCD head coming from different transformer layers. Training and testing are done on the synthetic dataset. Results are on the eval set. Note in Figure 1 that the layers are indexed from the top. λ in the multitask loss is kept as 1.0. Numbers are in $\%$.

input layer	WER	P	R	F1 (P&R)	Cov	Pur	F1 (Cov&Pur)
1	13.8	93.8	**97.2**	95.5	98.5	**98.9**	98.7
2	**13.2**	**96.1**	96.9	**96.5**	**99.0**	**98.9**	**98.9**
4	13.3	95.9	**97.2**	**96.5**	98.9	**98.9**	**98.9**
10	13.3	94.9	96.6	95.7	98.9	98.8	98.8

We experiment with different values of λ in the multitask loss to study its effects on the ASR and SCD performance. We observe that reducing λ from 1 to 0.5 and then to 0.1 improves ASR performance but degrades SCD metrics monotonically.

Speaker variabilities may hurt the ASR performance [17]. We attach the SCD head at the output of different transformer layers of the context network in the XLSR-53 model to mitigate the adverse effects of the SCD task on ASR and report the results in Table 3. There are 24 transformer layers in the XLSR-53 model and we start indexing from the top, shown in Figure 1. When the input to the SCD head is from layer 2, we see improvements in WER as well as speaker change metrics. Further, we give input to SCD head from much deeper layers but we either don't see any meaningful change in the results or a degradation. Our results support the hypothesis that SCD adversely affects ASR but its effect can be reduced by considering different embedding spaces from the XLSR-53 model as input to ASR and SCD heads.

Overall, it is clear from the results that multitask training notably enhances SCD performance. At the same time, it is better to use different embedding spaces for detecting speaker change and ASR. Depending upon the practical use-case, a combination of multitask and speaker token in reference text can be used to obtain better WER and speaker change metrics.

7. CONCLUSION

We introduce a multitask training method for joint speech recognition and speaker change detection. Experiments on the AMI corpus demonstrate that the proposed multitask model improves the speaker change detection metrics significantly without any degradation in ASR accuracy. We also show that it is better to use different embedding spaces from a self-supervised model like XLSR-53 for multitask ASR and speaker change tasks as compared to the outputs from the last layer. A combination of using speaker token augmented reference text for ASR training with speaker change detection classifier in a multitask fashion yields the best performance. Overall, compared to the baseline system when ASR is trained on text augmented with speaker-turn token, our proposed method yields an absolute improvement of 5.6% in precision, 0.7% in recall, and 3.2% in F1 score on eval set for SCD task with negligible impact on ASR accuracy.

8. REFERENCES

[1] Marek Hrúz and Zbyněk Zajíc, "Convolutional neural network for speaker change detection in telephone speaker diarization system," in *Proc. ICASSP*. IEEE, 2017, pp. 4945–4949.

[2] 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.

[3] Zbyněk Zajíc, Marie Kunešová, and Vlasta Radová, "Investigation of segmentation in i-vector based speaker diarization of telephone speech," in *Speech and Computer: 18th International Conference, SPECOM 2016, Budapest, Hungary, August 23-27, 2016, Proceedings 18*. Springer, 2016, pp. 411–418.

[4] Hervé Bredin, Claude Barras, et al., "Speaker change detection in broadcast tv using bidirectional long short-term memory networks," in *Interspeech 2017*. ISCA, 2017.

[5] Jitendra Ajmera, Iain McCowan, and Hervé Bourlard, "Robust speaker change detection," *IEEE signal processing letters*, vol. 11, no. 8, pp. 649–651, 2004.

[6] Fabio Valente, Deepu Vijayasenan, and Petr Motlicek, "Speaker diarization of meetings based on speaker role n-gram models," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4416–4419.

[7] Fabio Valente, Petr Motlicek, and Deepu Vijayasenan, "Variational bayesian speaker diarization of meeting recordings," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 4954–4957.

[8] Laurent El Shafey, Hagen Soltau, and Izhak Shafran, "Joint speech recognition and speaker diarization via sequence transduction," *arXiv preprint arXiv:1907.05337*, 2019.

[9] 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 *Proc. ICASSP*. IEEE, 2022, pp. 8077–8081.

[10] Jian Wu, Zhuo Chen, Min Hu, Xiong Xiao, and Jinyu Li, "Speaker change detection for transformer transducer asr," in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.

[11] 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 *Proc. ICASSP*. IEEE, 2023, pp. 1–5.

[12] Naoyuki Kanda, Yashesh Gaur, Xiaofei Wang, Zhong Meng, and Takuya Yoshioka, "Serialized output training for end-to-end overlapped speech recognition," *arXiv preprint arXiv:2003.12687*, 2020.

[13] Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Why does ctc result in peaky behavior?," *arXiv preprint arXiv:2105.14849*, 2021.

[14] Rui Zhao, Jian Xue, Jinyu Li, Wenning Wei, Lei He, and Yifan Gong, "On addressing practical challenges for rnn-transducer," in *2021 IEEE ASRU*. IEEE, 2021, pp. 526–533.

[15] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[16] Marie Kunešová and Zbyněk Zajíc, "Multitask detection of speaker changes, overlapping speech and voice activity using wav2vec 2.0," in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.

[17] Wei Zhou, Haotian Wu, Jingjing Xu, Mohammad Zeineldeen, Christoph Lüscher, Ralf Schlüter, and Hermann Ney, "Enhancing and adversarial: Improve asr with speaker labels," in *Proc. ICASSP*. IEEE, 2023, pp. 1–5.

[18] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "Flat-start single-stage discriminatively trained hmm-based models for asr," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 1949–1961, 2018.

[19] Hang Su, Danyang Zhao, Long Dang, Minglei Li, Xixin Wu, Xunying Liu, and Helen Meng, "A multitask learning framework for speaker change detection with content information from unsupervised speech decomposition," in *Proc. ICASSP*. IEEE, 2022, pp. 8087–8091.

[20] Jean Carletta, Simone Ashby, et al., "The ami meeting corpus: A pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.

[21] Petr Motlicek, Fabio Valente, and Philip N Garner, "English spoken term detection in multilingual recordings," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[22] Ivan Himawan, Petr Motlicek, David Imseng, Blaise Potard, Namhoon Kim, and Jaewon Lee, "Learning feature mapping using deep neural network bottleneck features for distant large vocabulary speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4540–4544.

[23] Daniel Povey, Arnab Ghoshal, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

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

[25] Hervé Bredin, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," in *Interspeech 2017*, Stockholm, Sweden, August 2017.

[26] Adam Paszke, Sam Gross, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[27] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[28] Apoorv Vyas, Srikanth Madikeri, and Hervé Bourlard, "Lattice-free mmi adaptation of self-supervised pretrained acoustic models," in *Proc. ICASSP*. IEEE, 2021, pp. 6219–6223.

[29] Daniel Povey, Gaofeng Cheng, et al., "Semi-orthogonal low-rank matrix factorization for deep neural networks.," in *Interspeech*, 2018, pp. 3743–3747.

[30] Srikanth Madikeri, Sibo Tong, et al., "Pkwrap: a pytorch package for lf-mmi training of acoustic models," *arXiv preprint arXiv:2010.03466*, 2020.

[31] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur, "Espresso: A fast end-to-end neural speech recognition toolkit," in *2019 IEEE ASRU*. IEEE, 2019, pp. 136–143.

[33] Yiwen Shao, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, "Pychain: A fully parallelized pytorch implementation of lf-mmi for end-to-end asr," *arXiv preprint arXiv:2005.09824*, 2020.

[34] "Kaldi s5b recipe results," `https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/RESULTS_ihm`.

[35] "Espnet egs2 recipe results," `https://github.com/espnet/espnet/tree/master/egs2/ami/asr1`.

[36] "K2-icefall recipe results," `https://github.com/k2-fsa/icefall/blob/master/egs/ami/ASR/RESULTS.md`.

[37] Dan Hendrycks and Kevin Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[38] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.