Interspeech 2022
18-22 September 2022, Incheon, Korea

Filler Word Detection and Classification: A Dataset and Benchmark

Ge Zhu[1*], *Juan-Pablo Caceres*[2], *Justin Salamon*[2]

[1]University of Rochester, [2]Adobe Research

ge.zhu@rochester.edu {caceres,salamon}@adobe.com

Abstract

Filler words such as 'uh' or 'um' are sounds or words people use to signal they are pausing to think. Finding and removing filler words from recordings is a common and tedious task in media editing. Automatically detecting and classifying filler words could greatly aid in this task, but few studies have been published on this problem to date. A key reason is the absence of a dataset with annotated filler words for model training and evaluation. In this work, we present a novel speech dataset, PodcastFillers, with 35K annotated filler words and 50K annotations of other sounds that commonly occur in podcasts such as breaths, laughter, and word repetitions. We propose a pipeline that leverages VAD and ASR to detect filler candidates and a classifier to distinguish between filler word types. We evaluate our proposed pipeline on PodcastFillers, compare to several baselines, and present a detailed ablation study. In particular, we evaluate the importance of using ASR and how it compares to a transcription-free approach resembling keyword spotting. We show that our pipeline obtains state-of-the-art results, and that leveraging ASR strongly outperforms a keyword spotting approach. We make PodcastFillers publicly available, in the hope that our work serves as a benchmark for future research.

Index Terms: filler word detection, speech disfluency, keyword spotting

1. Introduction

Speech disfluencies, such as filler words, stuttering, repetitions and corrections, are common in spontaneous speech [1]. Of all disfluencies, filler words, especially 'uh's and 'um's, are the most common [2]. For content creators working on, e.g., podcasts or video interviews, manually finding and editing filler words in video and audio recordings requires significant time and effort. Automatically detecting filler words accurately has the potential to significantly speed up speech content creation workflows. Such a filler word detection system must be able to both localize filler words in time and classify them correctly.

Previous work has focused on detecting and removing speech disfluencies from text transcripts [3–6], some also incorporating acoustic features [7]. In some cases, the transcripts are produced via Automatic Speech Recognition (ASR) [8–10]. In this scenario it is up to the ASR to transcribe the filler words, which requires training an ad-hoc ASR with filler words in its vocabulary. This is computationally intensive and challenging since ASR systems are often trained on spoken text corpora which do not contain any filler words, and thus cannot detect them reliably. Furthermore, adding a new filler word to the vocabulary would require re-training the ASR model.

More recently, several data driven methods have been proposed to detect speech disfluencies directly from audio in telephone conversations [11, 12] and naturalistic recordings [13–15]. Sheikh et al. proposed StutterNet [14], a time-delay neural network (TDNN) to classify repetitions, blocks, prolongations and interjections, the latter being another term for filler words. They used the UCLASS dataset [16], which is designed for stutter classification. This poses some challenges, since the dataset was recorded in a controlled environment, and exclusively contains speech by people who stutter. This, along with the small size of the dataset (∼4K sentences), means it is unclear whether the results would generalize to recordings of spontaneous speech more broadly. To tackle this issue, Kourkounakis et al. [15] expanded the training data by creating a synthesized stutter dataset, LibriStutter, by inserting repetitions or interjections in between non-stuttered speech from a subset of LibriSpeech [17]. The interjections, however, were taken from UCLASS, presenting the same aforementioned challenge. Salamin et al. [18] trained Hidden Markov Models to segment laughter, fillers, speech and silence from spontaneous speech on the SSPNet Vocalization Corpus, which is also a small-scale dataset. Lea et al. [12] created a large speech disfluency dataset, SEP-28K, from podcasts with people who stutter, and used it to build a stutter detector. As before, there is a generalization challenge given the audio data are specific to people who stutter. Also, the dataset is annotated at the clip level, so it does not provide precise timestamps for filler words and cannot be used to evaluate detection accuracy at a fine temporal resolution.

The closest study to our work is by Das et al. [1], who proposed a disfluency repair system aiming at removing filler words and long pauses. They trained a convolutional recurrent neural network (CRNN) for filler word segmentation (detection and classification) applied directly to audio recordings. They used two speech datasets, Switchboard speech data [19] with transcripts and Automanner [20]. A key limitation of the approach noted by the authors is that it was unable to distinguish filler words such as 'uh' or 'um' from real parts-of-speech, returning false-positives for actual words that sound similar to or contain filler words, such as "um-brella". Also, Kaushik et al. [13] found that the mismatch between training on telephony speech and testing on naturalistic recordings hurts filler classification accuracy. The test set for evaluating the methods presented by Das et al. [1] only contains 20 speech samples, once again making it hard to draw generalizable conclusions.

In this paper, we address the data scarcity challenge for filler word detection by creating the largest annotated dataset of filler words published to date, PodcastFillers, which we make publicly available online[1]. We propose an efficient workflow for generating annotation candidates in continuous speech recordings that leverages a robust Voice Activity Detection (VAD) model and an off-the-shelf ASR, and annotate over 85K filler word candidates. The resulting dataset spans 145 hours of speech from over 350 speakers coming from 199 public podcast episodes, and has 35K annotated filler words and 50K annotations of other speech events that are common in podcasts

*This work was performed during an internship at Adobe Research.

[1]podcastfillers.github.io

10.21437/Interspeech.2022-10992

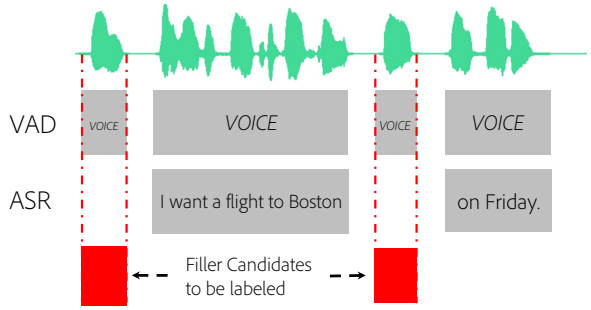

Figure 1: *Filler word candidate generation pipeline: non-linguistic filler word candidates are identified at times where VAD is activated while ASR is not.*

such as laughter, breaths, and repetitions. It also includes the ASR transcriptions we obtained for all the episodes. Using PodcastFillers, we train a filler classifier similar to a keyword spotting approach, and present an end-to-end pipeline that leverages VAD, ASR, and the classifier to perform filler word detection and classification. We compare our proposed pipeline to two baselines and show that it yields state-of the-art results. We hope it serves as a robust benchmark for future research.

2. The PodcastFillers Dataset

2.1. Podcast curation

We manually curated 199 gender-balanced, English-language podcast episodes from SoundCloud[2], totaling 145 hours of speech audio from over 350 speakers sampled at 44.1 kHz. We searched for episodes using the 24 topic categories defined in [21] to include a variety of topics and styles, and selected episodes from different shows or shows with guest speakers to ensure a diversity of speakers.

2.2. Filler word annotation pipeline

Listening to the entire dataset to label fillers would be highly inefficient. Instead, we propose an annotation pipeline that leverages a commercial ASR system[3] and a VAD model to generate filler candidates for annotation, depicted in Fig. 1.

As previously noted, ASR systems typically do not transcribe non-lexical fillers in spontaneous speech. We make use of this drawback to identify possible filler word locations: non-lexical fillers such as 'uh' and 'um' will trigger the VAD model, but appear as silent gaps in the ASR output. These regions where VAD activates but ASR does not are candidate locations for fillers. Using this approach we identified 85K candidates in PodcastFillers. Since the candidates may contain other sounds such as breaths, laughter, music, or even words (due to ASR errors), they require manual verification, which we obtained via crowdsourcing using a custom-built annotation interface.

Voice Activity Detection (VAD) model and data: For detecting candidate fillers, we need a VAD model that outputs predictions at a fine temporal resolution (100 Hz) to precisely locate the temporal boundaries of speech regions. It also needs to be robust to various background and foreground noises in podcasts such as music and non-speech sounds (e.g., fan noise).

[2] www.soundcloud.com
[3] www.speechmatics.com

We achieve this fine temporal resolution by computing input acoustic features at a 10 ms hop size, such that we can slide the trained model over the audio at this temporal resolution. To ensure robustness, we need to combine a generalizable ML model with a varied training set containing various background and foreground noises at different signal-to-noise ratios (SNR). We create a new labeled speech dataset for VAD by programatically combining recordings of clean speech with music and noise using the Scaper soundscape mixing software [22].

We generate frame-level (10 ms) VAD annotations by computing the audio amplitude from clean speech recordings sourced from the Librispeech-100 [17] and VCTK [23] datasets, labeling regions below a 19 dB threshold relative to the peak amplitude of the normalized signal as silent. Then we programmatically mix the clean speech clips with background music and environmental sound from the strongly labeled subset [24] of AudioSet [25]. To test our VAD model, we keep a disjoint test set of audio source material using 8% of the speakers in VCTK, the test partition of Librispeech, and 5% of sound events from the AudioSet subset. We generate 300,000 training mixtures from the training source material and 10,000 test mixtures from the test source material. Scaper allows us to control the SNR range and distribution in the mixtures. Preliminary experiments showed the performance of the VAD model, in terms of producing filler candidates when combined with ASR, was sensitive to the SNR range. We empirically settled on a speech SNR range of $[12, 22]$ dB relative to background noise, $[-3, 17]$ dB relative to foreground noise and $[-6, 14]$ dB relative to music.

We compute log-scaled mel-spectrograms (log-mel) as input to the VAD model using Librosa [26]. We use 64 mel bins, and a purposely short window of 25ms and a hop size of 10ms, to support inference at a high temporal resolution. We adopt a Convolutional Recurrent Neural Netowrk (CRNN) architecture that has been shown to be robust for VAD in complex environments with noise [27], but remove the recurrent layer to improve run-time performance. We found this change does not impact model accuracy. Our trained VAD model obtained Precision/Recall of 0.93/0.92 respectively on the test split of our mixed dataset. Once our VAD model was trained, we used it in combination with the ASR to produce filler candidates. To minimize the chance of missing soft fillers, we set a lenient VAD activation threshold of 0.1 (as opposed to the standard 0.5 out of $[0, 1]$). We found the majority of candidates to have a duration in the 150-400 ms range. For candidates shorter than 150 ms it was hard to determine by ear whether they were actual filler words or other sounds, so we decided to remove them prior to labeling. Similarly, we removed candidates longer than 2 s, which were rare and not representative of our target use case.

Labeling filler candidates: Based on an initial audition of a sample of candidates, we identified a set of filler and non-filler classes for our labeling task. The labels, along with the final number of annotations per label (in parentheses), are: For fillers, 'uh' (17907), 'um' (17078), 'you know' (668), 'like' (157), and 'other' (315). 'Like' and 'you know' occurred rarely in our candidate set, when the ASR failed to transcribe them. For non-fillers, 'laughter' (6623), 'breath' (8288), 'agreement sound' (3755, e.g., 'mmm' or 'uh-huh'), 'regular words' (12709), 'repetitions' (9024), 'simultaneous speakers' (1484), 'music' (5060) and 'noise' (2735). The first three non-filler labels represent voice sounds that aren't fillers. The next three are caused by ASR errors or intentional omissions, and the final two are caused by VAD false-positives.

The candidates were presented to crowd workers for annotation. Each filler candidate was positioned at time 3 sec inside

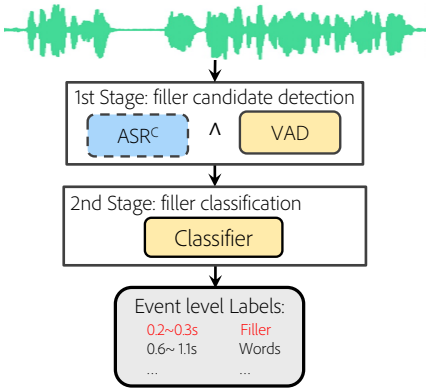

Figure 2: *Proposed two-stage filler detection and classification.*

a 5 sec clip (for context), and highlighted in the interface. Annotators had to determine whether the highlighted candidate was a filler word or not, and based on that select one of the five filler labels or eight non-filler labels. Each candidate was annotated by two people, or three when the first two disagreed. Out of all candidates labeled 'uh' or 'um' in the dataset, 98.4% and 96.4% respectively had at least two annotators agree on the label.

3. Filler Detection Pipeline

We propose a filler detection pipeline with two variants: the first leverages ASR, while the second does not, which is relevant for deployment scenarios where ASR is not available. The pipeline is depicted in Fig. 2. In the first stage, the input audio is passed through the VAD model to find voice regions. The first pipeline variant also runs the audio through ASR to discard regions with transcribed words. The second variant passes straight to the next stage. In the second stage, the remaining candidate time regions are passed through a classification model that produces labeled events with a start time, end time, and a label. Moving forward, we shall refer to the first pipeline as AVC-FillerNet (for ASR + VAD + Classifier), and the second as VC-FillerNet (no ASR). By skipping the ASR, VC-FillerNet is computationally lighter, but runs the risk of detecting parts of actual words as fillers [1].

Our goal is to train a robust multi-class classifier to detect fillers given a short snippet of audio. Given the label distribution in PodcastFillers, we opted to discard labels with 3K or less annotations and consolidate other labels, producing five new labels, each with ample training data: 'filler' ('uh'+'um'), 'words' ('regular words' and 'repetitions'), 'laughter', 'music', and 'breath'. Ultimately, we only care about the detection accuracy for the 'filler' class, and we expect this consolidation to lead to a more robust classifier. In Section 5 we also evaluate our ability to classify 'uh' and 'um' as two separate classes.

We use wav2vec [28] embeddings computed with a 10 ms hop size as input to the model. Wav2vec was pretrained on over 960 hours of speech, providing a robust representation for classification of speech-like sounds. During training we apply time and "frequency" masking to the embeddings via SpecAugment [29], and optimize a cross entropy loss.

Since our goal is to detect specific short utterances in an audio stream, the task can be viewed as a keyword-spotting (KWS) problem where our keyword is the joint set of 'uh' and 'um'. With this in mind, we adapt a lightweight KWS model backbone architecture, TC-ResNet8 [30], for efficient classification. TC-ResNet8 only has around 100k parameters, making it suitable for low-latency inference. It applies 1D convolutions along the temporal axis and spans the entire frequency range in every layer, achieving strong performance even with a small number of layers. Because the filler candidates in AVC-FillerNet are normally short segments, we can train an *event classifier* to directly predict the event label for the entire input segment. On the contrary, for VC-FillerNet, the filler candidates are usually long sequences of voice, so we train a *frame classifier* to predict frame-level labels at a fine temporal resolution, e.g., every 100 ms. To get frame-level predictions, we adapt the TC-ResNet8 backbone by adding an LSTM layer. Similar to Filler-CRNN [1], we then group contiguous frames with the same predicted label into an event. The final output of both the event-level classifier and frame-level classifier are discrete events with a start time, end time, and a label. We compare the two approaches as part of our ablation study.

4. Experimental Design

4.1. Data split and training

For our experiments, we split the PodcastFillers dataset into train, validation, and test sets with 173, 6, 20 episodes respectively, while ensuring each subset remains gender-balanced. The audio is downsampled from 44.1 kHz to 16 kHz for computational efficiency. We train our proposed models on the training set, tune hyper-parameters and the VAD threshold on the validation set, and report performance on the test set. To train the event classifier, AVC-FillerNet, we use 1 s input clips with the labeled filler candidate placed at the center of the clip. The model produces a single prediction for the input, which is compared to the ground truth label. To train the frame classifier, VC-FillerNet, we use 1 s input clips where the filler candidate can appear anywhere in the clip. The model produces per-frame (10) predictions that are compared to the ground truth events (which have start/end times) frame-by-frame during training.

4.2. Baselines

We compare our systems with two strong baselines: a neural-network-based method, Filler-CRNN [1], and a forced-aligner-based method, Gentle [31]. The input to Filler-CRNN is log-mel with 128 bins computed from 1 s clips. The original Filler-CRNN architecture yielded weak performance in our experiments, so we fine tune it (number of layers, kernel size, pooling) on PodcastFillers for a stronger baseline. With Gentle, we first apply pre-trained Kaldi [32] acoustic models developed on the Fisher English corpus [33] to generate syllable tokens. We compare the tokens with the ASR transcript: filler words are detected if the inconsistent regions between the two are re-aligned by inserting 'um' or 'uh' into the transcript.

4.3. Evaluation metrics

We compute segment-based and event-based metrics (Precision, Recall, F1) using sed_eval [34] for the 'filler' class, to evaluate detection accuracy and localization accuracy respectively. Segment-based metrics map the system output and ground truth to a fixed time grid for comparison. Event-based metrics compare the estimated sound events and the ground truth events directly. A predicted event is considered a true positive if it overlaps with a ground truth event that has the same label *and* its onset and offset are within a threshold (slack) from the reference event's onset and offset (200 ms in this work).

4.4. Ablations

We run ablation studies to understand the impact of each of the two stages of our proposed pipeline, VAD and the filler classifier. For VAD, we vary the activation threshold from 0.1–0.9: the lower the threshold the more candidates will be passed to the second stage. For the classifier, we compare different input features (log-mel with 64 bins and wav2vec) and architectures (event classifier and frame classifier). We evaluate both the AVC-FillerNet and VC-FillerNet pipelines in all ablations. The output events from each model are converted to frame-level likelihoods, which are compared to the reference annotations to produce Precision-Recall (PR) curves, which elucidate the trade-off between precision and recall.

5. Results

5.1. Ablation studies

We start by analyzing the influence of the VAD activation threshold, using the PodcastFillers validation set, and wav2vec as the input feature. The results are shown in Fig. 3(a). The best PR-curve is obtained using the lowest threshold (0.1), and as we increase the threshold there is a notable decrease in recall. Interestingly, the precision remains consistent regardless of the VAD activation threshold. This suggests that the filler classifier is robust at rejecting false positives with lower VAD likelihoods, and so a low VAD threshold maximizes recall by ensuring we do not miss soft filler words, without compromising on precision.

Following this, we fix the VAD threshold to 0.1 in the second ablation study where we compare different input features and classifier backbones, shown in Fig. 3(b). Wav2vec consistently outperforms log-mel as the input feature, confirming that this model, trained on a large speech corpus, yields a discriminative representation for filler word classification. For the AVC-FillerNet pipeline, the event classifier is only marginally better than the frame classifier. In contrast, since the VC-FillerNet pipeline cannot leverage ASR to determine the precise timing of filler candidates, the frame classifier outperforms the event classifier in this pipeline due to its superior temporal accuracy.

Most importantly, we see that across both ablation studies, AVC-FillerNet clearly outperforms VC-FillerNet. By leveraging ASR, AVC-FillerNet produces tight temporal boundaries around filler candidates, and dramatically reduces the number of candidates passed to the classifier. This reduces the chances of producing false positives, leading to a boost in precision.

5.2. Comparison to baselines

We compare AVC-FillerNet and VC-FillerNet with two baselines, Filler-CRNN and Gentle (described in Sec 4.2), and report results the PodcastFillers test set as shown in Tab. 1. For our pipelines, we use the optimal VAD threshold of 0.1 as determined by the ablation study on the validation set. We see that AVC-FillerNet significantly outperforms all the other systems for all metrics. Gentle yields higher precision than VC-FillerNet and Filler-CRNN, but has the lowest recall among all the systems. We speculate this may be improved by leveraging acoustic models trained with filler words in Gentle.

5.3. Filler detection & classification with fine granularity

Finally, we evaluate our systems' ability to separately detect 'uh' and 'um' filler words. We re-train the classifier with the two labels as separate classes, and evaluate our systems on the PodcastFillers test set, as shown in Tab. 2. We see that 'um' is easier to classify, especially for VC-FillerNet. We speculate that this is because 'um's are typically longer than 'uh's, providing the classifier with more signal to leverage for inference.

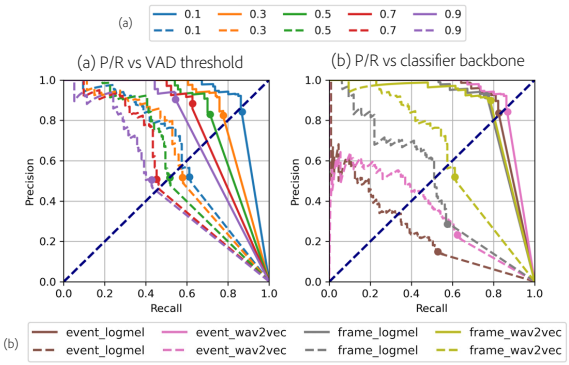

Figure 3: *Frame level P/R vs (a) VAD threshold and (b) classifier backbone. 'event' stands for event classifier and 'frame' for frame classifier, solid lines are AVC-FillerNet, dashed lines are VC-FillerNet. Dotted position is the classifier threshold at 0.5.*

Table 1: *Segment- and event-based evaluation results (%) for our proposed systems and baselines for filler word detection.*

System	Segment			Event		
	P	R	F1	P	R	F1
AVC-FillerNet (Ours)	**93.0**	**95.4**	**94.2**	**91.7**	**94.0**	**92.8**
VC-FillerNet (Ours)	71.6	71.0	71.3	66.0	76.9	71.0
Filler-CRNN [1]	56.4	70.3	62.6	37.5	78.3	50.7
Gentle [31]	78.4	64.8	71.0	77.0	64.9	70.4

Table 2: *Segment- and event-based F1 measure (%) results for separately detecting 'uh' and 'um' with our proposed systems.*

System	'Um'		'Uh'	
	Segment	Event	Segment	Event
AVC-FillerNet	92.5	91.0	85.0	84.3
VC-FillerNet	75.2	75.9	57.0	57.1

6. Conclusion

In this work we presented PodcastFillers, a large dataset of podcasts with annotated filler words. The dataset was created by boostrapping a VAD model and a commercial ASR system to generate filler candidates that were annotated via crowdsourcing. We proposed ASR-based and ASR-free filler detection and classification pipelines, AVC-FillerNet and VC-FillerNet. Our experiments showed that AVC-FillerNet achieves state-of-the-art results, significantly outperforming existing filler word detection systems, and that leveraging ASR outperforms a keyword spotting approach for filler word detection. Through ablation studies, we evaluated the impact of our design choices on system performance. We hope the PodcastFillers dataset, our proposed filler detection and classification pipeline, and our experimental results serve as a benchmark for future research.

7. References

[1] S. Das, N. Gandhi, T. Naik, and R. Shilkrot, "Increase apparent public speaking fluency by speech augmentation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6890–6894.

[2] K. Womack, W. McCoy, C. O. Alm, C. Calvelli, J. B. Pelz, P. Shi, and A. Haake, "Disfluencies as extra-propositional indicators of cognitive processing," in *Proceedings of the workshop on extra-propositional aspects of meaning in computational linguistics*, 2012, pp. 1–9.

[3] N. Bach and F. Huang, "Noisy bilstm-based models for disfluency detection." in *INTERSPEECH*, 2019, pp. 4230–4234.

[4] F. Wang, W. Chen, Z. Yang, Q. Dong, S. Xu, and B. Xu, "Semi-supervised disfluency detection," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 3529–3538.

[5] P. Jamshid Lou, P. Anderson, and M. Johnson, "Disfluency detection using auto-correlational neural networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP2018)*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 4610–4619.

[6] S. Wang, W. Che, Q. Liu, P. Qin, T. Liu, and W. Y. Wang, "Multi-task self-supervised learning for disfluency detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9193–9200.

[7] V. Zayats and M. Ostendorf, "Giving attention to the unexpected: Using prosody innovations in disfluency detection," in *Proceedings of NAACL-HLT*, 2019, pp. 86–95.

[8] J. Ferguson, G. Durrett, and D. Klein, "Disfluency detection with a semi-markov model and prosodic features," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 257–262.

[9] H. Hassan, L. Schwartz, D. Hakkani-Tür, and G. Tur, "Segmentation and disfluency removal for conversational speech translation," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[10] P. A. Heeman, R. Lunsford, A. McMillin, and J. S. Yaruss, "Using Clinician Annotations to Improve Automatic Speech Recognition of Stuttered Speech," in *Proc. Interspeech 2016*, 2016, pp. 2651–2655.

[11] R. Gupta, K. Audhkhasi, S. Lee, and S. S. Narayanan, "Paralinguistic event detection from speech using probabilistic time-series smoothing and masking." in *Interspeech*, 2013, pp. 173–177.

[12] C. Lea, V. Mitra, A. Joshi, S. Kajarekar, and J. P. Bigham, "Sep-28k: A dataset for stuttering event detection from podcasts with people who stutter," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6798–6802.

[13] L. Kaushik, A. Sangwan, and J. H. Hansen, "Laughter and filler detection in naturalistic audio." International Speech and Communication Association, 2015.

[14] S. A. Sheikh, M. Sahidullah, F. Hirsch, and S. Ouni, "Stutternet: Stuttering detection using time delay neural network," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 426–430.

[15] T. Kourkounakis, A. Hajavi, and A. Etemad, "Fluentnet: End-to-end detection of stuttered speech disfluencies with deep learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2986–2999, 2021.

[16] P. Howell, S. Davis, and J. Bartrip, "The university college london archive of stuttered speech (uclass)," *Journal of Speech, Language, and Hearing Research*, vol. 52, pp. 556–569, 2009.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[18] H. Salamin, A. Polychroniou, and A. Vinciarelli, "Automatic detection of laughter and fillers in spontaneous mobile phone conversations," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 4282–4287.

[19] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1. IEEE Computer Society, 1992, pp. 517–520.

[20] M. I. Tanveer, R. Zhao, K. Chen, Z. Tiet, and M. E. Hoque, "Automanner: An automated interface for making public speakers aware of their mannerisms," in *Proceedings of the 21st International Conference on Intelligent User Interfaces*, 2016, pp. 385–396.

[21] G. Chen, S. Chai, G.-B. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, M. Jin, S. Khudanpur, S. Watanabe, S. Zhao, W. Zou, X. Li, X. Yao, Y. Wang, Z. You, and Z. Yan, "GigaSpeech: An Evolving, Multi-Domain ASR Corpus with 10,000 Hours of Transcribed Audio," in *Proc. Interspeech 2021*, 2021, pp. 3670–3674.

[22] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 344–348.

[23] C. Veaux, J. Yamagishi, K. MacDonald *et al.*, "Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," 2016.

[24] S. Hershey, D. P. Ellis, E. Fonseca, A. Jansen, C. Liu, R. C. Moore, and M. Plakal, "The benefit of temporally-strong labels in audio event classification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 366–370.

[25] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[26] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015, pp. 18–25.

[27] Y. Chen, H. Dinkel, M. Wu, and K. Yu, "Voice Activity Detection in the Wild via Weakly Supervised Sound Event Detection," in *Proc. Interspeech 2020*, 2020, pp. 3665–3669.

[28] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-Training for Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 3465–3469.

[29] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[30] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019.

[31] R. Ochshorn and M. Max Hawkins, "Gentle: A robust yet lenient forced aligner built on kaldi." https://lowerquality.com/gentle/.

[32] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

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

[34] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.