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. A key reason is the absence of a dataset with annotated filler words for 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, and hope 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, in some cases transcripts produced via Automatic Speech Recognition (ASR) [3–6]. 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 recordings [7–9]. Sheikh et al. proposed StutterNet [7], 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 [10], 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. [8] 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 [11]. The interjections, however, were taken from UCLASS, presenting the same aforementioned challenge. Lea et al. [9] 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. Additionally, 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 for filler word segmentation (detection and classification) applied directly to audio recordings. They used two speech datasets, Switchboard speech data with transcripts [12] and Automanner [13]. 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 part-of-speech, thus returning false-positive detection for actual words that sound similar to or contain filler words, such as "um-brella". Also, the test set for evaluating the different methods presented in [1] contained only 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 such as laughter, breaths, and repetitions. It also includes the ASR transcriptions we produced 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.

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

[1]The dataset will be made public upon paper publication.

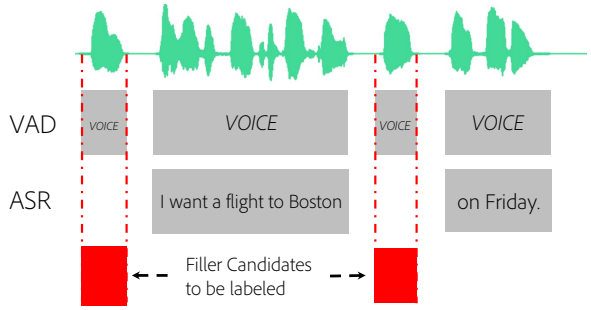

Figure 1: *Filler word candidate generation pipeline: non-linguistic filler word candidates are generated at times where VAD is activated while ASR is not.*

2. The PodcastFillers Dataset

2.1. Podcast curation

We manually curated 199 CC-licensed, 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 [14] 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 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).

We achieve the 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 adopt a deep neural network (DNN) architecture that has been shown to be robust for VAD in complex environments with noise [15]. To ensure good generalization to unseen noise, 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 [16]. We generate frame-level (10 ms) VAD annotations by computing the audio amplitude from clean speech recordings sourced from the Librispeech-100 [11] and VCTK [17] datasets, labeling regions below a dB threshold relative to the peak amplitude of the normalized signal as silent. Then we programmatically mixed the clean speech clips with background music and environmental noise from Audioset [18]. Using Scaper allowed 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 generated 300,000 mixtures to train our VAD model.

We compute log-scaled mel-spectrograms (log-mel) as input to the VAD model using Librosa [19]. 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 adapt a convolutional recurrent neural network (CRNN) architecture proposed for VAD [15] by removing the recurrent layer for improved runtime 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 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 noises, so we decided to remove them prior to labeling. Similarly, we removed candidates above 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 (rounded, in parentheses), were: For fillers, 'uh' (18K), 'um' (17K), 'you know' (<1K), 'like' (<.5K), and 'other' (<.5K). 'Like' and 'you know' occurred rarely in our candidate set, when the ASR failed to transcribe them. For non-fillers, 'laughter' (7K), 'breath' (5K), 'agreement sound' (4K, e.g., 'mmm' or 'uh-huh'), 'regular words' (13K), 'repetitions' (9K), 'simultaneous speakers' (1.5K), 'music' (8K) and 'noise' (3K). The first three 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.

To annotate the candidates we presented them to crowd workers within a 5 sec. clip for context. The filler candidate was always positioned at time 3 sec. and highlighted in the interface. Annotators had to determine whether the highlighted candidate was a filler word or not, and depending on this answer select one of the five filler labels or eight non-filler labels. Each filler candidate was annotated by two people, and a third broke the tie in case the first two disagreed. The results shows that for fillers 'uh' and 'um' the annotators reached a verdict on up to 98.4% and 96.4%, respectively.

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

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

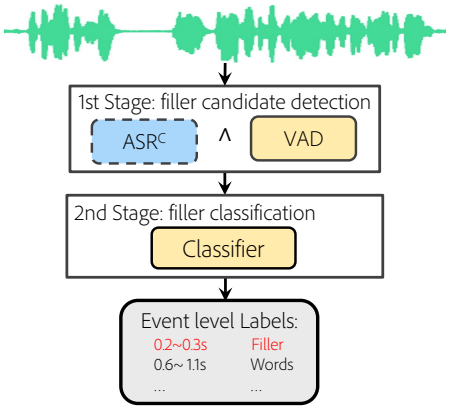

Figure 2: *Proposed two-stage filler detection and classification.*

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].

3.1. The filler classifier

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 compute wav2vec [20] embeddings with an equivalent 10ms hop size as model input. 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 [21] as data augmentation, and optimize a cross entropy loss.

Since our goal is to detect specific short utterances in an audio stream, it can be viewed as a keyword-spotting (KWS) task 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 [22], for efficient classification. TC-ResNet8 has only around 100k parameters, making it low latency for prediction. It applies 1D convolution in 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. Similar to Filler-CRNN [1], we then group the frame-level predictions into events via simple post-processing. To get frame-level predictions, we adapt the TC-ResNet8 backbone by adding an LSTM layer. 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

First, we first present the dataset preparation steps for training the filler classifier. Then we introduce different baseline systems and evaluation metrics to benchmark. Lastly, we conduct ablation studies to measure the contributions from individual subsystems.

4.1. Dataset

We first downsample the original 199 podcasts from 44.1kHz to 16kHz to reduce computation cost. Then we manually split the preprocessed podcasts into train, validation and test set keeping gender-balance: we train the event classifier and the frame classifier on the training partition, tune hyper-parameters including VAD threshold and backbones on the validation set and finally compare the final performance on the test set. When training the event classifier, we use a 1 second speech segment containing the filler candidate in the middle (ground truth label). To train the frame classifier, we also use a 1 second speech segment but with event based interval labels instead.

4.2. Baseline systems

For the baseline systems, we compare our systems with two strong baselines in filler word detection including a neural network based method Filler-CRNN [1] and a forced aligner based method Gentle [23]. For Filler-CRNN, we extracted log mel of 128 bins from 1 second speech following the original feature extraction steps. Filler-CRNN original architecture would yield relatively weak performance in our experiments, so we fine tune its hyperparameters for a stronger baseline.

With Gentle, we first apply pre-trained Kaldi [24] acoustic models developed on the Fisher English corpus [25] to generate syllable tokens as reference. We then compare this reference with fluent transcripts generated by Speechmatics. Fillers are detected if the inconsistent regions between the two are realigned by inserting 'um' or 'uh'.

4.3. Evaluation metrics

We compare segment-based and event-based F1 measure using sed_eval [26] on the 'filler' class to evaluate detection accuracy and location accuracy respectively.

Segment-based metrics compare system outputs in a fixed time grid against ground truth labels, which is equivalent to comparisons at the frame level. To use this metric, we need to convert all the system outputs into frame level labels.

Event-based metrics, on the other hand, treat fillers as sound events and jointly evaluate its start time, end time and label. A predicted event is treated as 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

To measure the impact of VAD module and classifier, we run two experiments on both AVC-FillerNet and VC-FillerNet. For the VAD system, the threshold controls how much voice would

go through the first stage: some filler candidates will be filtered out when increasing VAD threshold. For the classifier ablations, we first compare different input features including the same ones used earlier, log mel of 64 mel bins and wav2vec extracted vectors. Finally, we compare the frame classifier and event classifier backbones.

In ablation comparisons, we convert the raw output from the event classifier into frame level confidence and then compute precision and recall (PR) curves rather than single precision and recall value described in Section.4.3. PR curves can show a trade-off between precision and recall at different thresholds, and it's also informative when class labels are imbalanced.

5. Results

We first present two experiments to see how each module influence filler detection performance. Then we compare our best performing system with two existing systems. Lastly, we extend our model with a finer granularity splitting the fillers classes in two. For this, we retrain with separate classes of 'um' and 'uh' instead of one joint 'filler' class.

5.1. Ablation studies

We start by analyzing the influence of the VAD thresholds on PodcastFillers validation set using wav2vec as input feature. We compare frame wise PR curve varying VAD threshold from 0.1 to 0.9 with 0.2 as step size. The results are shown in Fig. 3(a). We observe that when the threshold increases there's a significant decrease in recall that doesn't compromising precision (the PR curve gradually shift to the left). When increasing the VAD threshold, the fillers of low voice confidence will probably be filtered out at the first stage, resulting in the decrease of recall. At the same time, fillers of high voiced confidence would not be influenced by the threshold increase, and consequently, the precision would not change much.

Based on the above observation, we fixed the VAD threshold to 0.1 to compare different input features and filler classifier backbones (Fig. 3(b)). We observe that wav2vec consistently outperforms log mel as input feature, showing the superiority of wav2vec in speech related tasks. We also see that for AVC-FillerNet, the event classifier is only marginally better than the frame classifier. In VC-FillerNet, however, the frame classifier significantly outperforms the moving event classifier. Instead of predicting labels locally in frame classifier, event classifier makes use of the context information from the whole second, which may create more false positives in the fluent speech area.

It is important to note that in both plots in Fig. 3, AVC-FillerNet significantly outperform VC-FillerNet. This is explained by the accurate boundaries of fluent speech output from ASR system largely narrow down the filler candidate region.

5.2. System comparison

We compare our filler detection systems with two other baselines, Filler-CRNN and Gentle (described in Sec 4.2). Tab. 1 shows the comparison on the test set of our two systems and the two baselines. As discussed in Section. 5.1, we pick a VAD threshold of 0.1 that achieved optimal precision and recall for both AVC-FillerNet and VC-FillerNet. We can see that AVC-FillerNet significantly outperforms all the other systems in every metric. We can also observe that Gentle has higher precision than VC-FillerNet and Filler-CRNN but has the lowest recall among all the systems. We speculate that this can be improved in Gentle with specific filler words training.

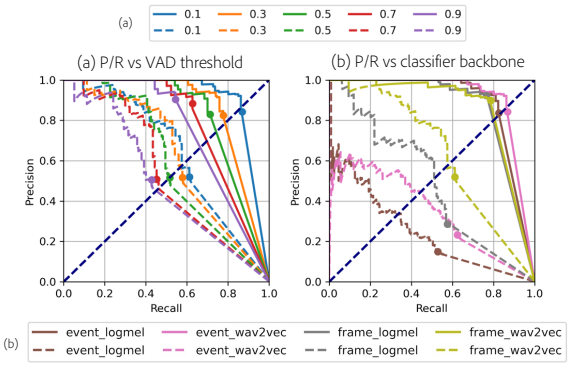

Figure 3: *Frame level P/R vs VAD threshold and classifier backbone. In each figure, 'event' stands for event classifier and 'frame' stands for frame classifier, solid lines are AVC-FillerNet, dash lines are VC-FillerNet. Dotted position is the classifier threshold at 0.5.*

Table 1: *A comparison of several baselines using segment based and event based precision (%) and recall (%)*

System	Segment			Event		
	F1	P	R	F1	P	R
AVC-FillerNet (Ours)	**94.2**	**93.0**	**95.4**	**92.8**	**91.7**	**94.0**
VC-FillerNet (Ours)	71.3	71.6	71.0	71.0	66.0	76.9
Filler-CRNN [1]	62.6	56.4	70.3	50.7	37.5	78.3
Gentle [23]	71.0	78.4	64.8	70.4	77.0	64.9

5.3. Filler detection with fine granularity

We perform an additional evaluation splitting our filler class into separate 'um' and 'uh' classes. We compute F1 for both segment and event based metrics (Tab. 2). We observe that 'um' is easier to classify, especially in VC-FillerNet. We speculate that this is because 'um's are usually longer than 'uh's, giving the classifier more "time" to make a decision.

Table 2: *Evaluation of segment based and event based F1 measure (%) on separate filler class from 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 created a large filler word dataset, PodcastFillers, by bootstrapping our VAD system with a commercial ASR system. We proposed ASR-based and ASR-free filler detection and classification pipelines, AVC-FillerNet and VC-FillerNet. We measured detection and location accuracy of our systems using segment based and event based precision and recall. Experiments showed that AVC-FillerNet achieved state-of-the-art results of over 90% in both metrics, significantly outperforming existing filler words detection systems. We also showed that that leveraging ASR strongly outperformed keyword spotting approaches.

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] 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.

[4] 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. [Online]. Available: https://www.aclweb.org/anthology/D18-1490.pdf

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-0995

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

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