CAM: Uninteresting Speech Detector

Weiyi Lu, Yi Xu, Peng Yang, Belinda Zeng

Amazon.com

{weiyilu,yxaamzn,pengya,zengb}@amazon.com

Abstract

Voice assistants such as Siri, Alexa, etc. usually adopt a pipeline to process users' utterances, which generally include transcribing the audio into text, understanding the text, and finally responding back to users. One potential issue is that some utterances could be devoid of any interesting speech, and are thus not worth being processed through the entire pipeline. Examples of uninteresting utterances include those that have too much noise, are devoid of intelligible speech, etc. It is therefore desirable to have a model to filter out such useless utterances before they are ingested for downstream processing, thus saving system resources. Towards this end, we propose the Combination of Audio and Metadata (CAM) detector to identify utterances that contain only uninteresting speech. Our experimental results show that the CAM detector considerably outperforms using either an audio model or a metadata model alone, which demonstrates the effectiveness of the proposed system.

Index Terms: audio event detection, acoustic scene classification

1. Introduction

The past decade has witnessed a boom of artificial intelligence applications, among which voice assistant is one such example. It first came into the public eye back in October 2011, when Apple released Siri. Then in November 2014, Amazon debuted the echo device and Alexa. Since then, these applications have been growing exponentially in terms of the number of devices, customers, and features (skills) supported. Under the hood, voice assistants rely on a pipeline of machine learning (ML) models for a variety of tasks, including transcribing the audio into text, understanding the text, and finally responding back to users.

A potential issue here is that, some utterances could be devoid of any interesting speech, and are thus not worth being processed through the entire pipeline. There are various reasons why an utterance might be considered uninteresting. Some examples include:

- The utterance is devoid of intelligible speech, or the audio is garbled.
- The utterance has too much noise.
- The utterance is media playing at a very high volume.

These utterances are not false accepts, that is, the device is activated due to the detection of some key phrase (i.e. the wake word) or the talk button being pressed. It is just that the utterance that follows the activation does not contain any interesting speech. Having to deal with such uninteresting utterances results in wasted system resources and reduces the efficiency of the pipeline. Motivated by this, the goal of this paper is to build a model to detect utterances that contain only uninteresting speech, so that we can prevent them from being processed in downstream tasks.

Towards this end, we draw lessons from previous works in areas such as acoustic scene classification and audio event detection, and build a neural network model that identifies uninteresting utterances based on acoustic features extracted from raw audio data. At the same time, since the same acoustic features are also utilized by the automatic speech recognition (ASR) model to transcribe the utterance, it only adds minimal overhead if we also pass the acoustic features through the ASR system, and the benefit in return is that we can retrieve the metadata generated by the ASR system for the utterances, such as recognition confidence. This allows us to train, in addition to the audio model, a metadata model that classifies based on both the ASR metadata and other variables such as audio duration. Finally, we combine the power of the two models together through an ensemble model, which we call the Combination of Audio and Metadata (CAM) detector. Experimental results show that the CAM detector achieves a significant performance boost over both the audio model and the metadata model, which demonstrates the effectiveness of this proposed system.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. We present our proposed solution in Section 3 and demonstrate its efficacy in Section 4 through experimental results. Finally, Section 5 draws conclusion and points out future work.

2. Related Work

As already mentioned, the reason why an utterance is considered uninteresting ranges from the utterance being devoid of intelligible speech, having too much noise, or being a media speech, etc. Given these scenarios, the problem we are trying to solve here is similar to acoustic scene classification (ASC) and audio event detection (AED). We do note that there is one distinction between our problem and the conventional applications, namely most existing works focus on one specific type of audio event such as media speech [1], while we deal with several scenarios at once. Despite this, our approach still draws lessons from ASC and AED, and therefore we review the related works in both of these areas below.

ASC aims to recognize the type of environment where a given recording takes place. [2] offer a tutorial on ASC with a particular emphasis on computational algorithms designed to perform this task automatically. Authors of [3] propose to combine LSTM and CNN in parallel to tackle the task, and the proposed method is shown to be able to to learn complementary information from both networks.

As for AED, [4] propose to solve the task by leveraging multi-channel information. [5] propose a parallel CNN-GRU-FC architecture which lays the foundation for later works including [1], where the authors adopt a similar model architecture to detect media speech, and their model uses multi-channel information as well. Authors of [6] design an end-to-end CNN for AED. In particular, they propose an attention-based CNN which weights the contribution of each feature, and they motivate the formulation of this model from a probabilistic perspective. [7] propose to use CNN to extract features from spectrograms. And finally, [8] propose an end-to-end CNN system, which adopts small filters as inspired by VGGNet [9]. They also propose a novel data augmentation technique that boosts the models performance.

For both ASC and AED, there are also works that explore methods other than CNNs and LSTMs. For example, [10] study the usefulness of various matrix factorization methods for learning features for ASC. And authors of [11] present a system for AED using a hidden Markov model (HMM).

3. Our Proposal CAM

As seen in the introduction, uninteresting utterances can generally be identified with some unique acoustic characteristics, e.g. the utterance being unintelligible or garbled, or being media speech, etc. We believe these characteristics will manifest themselves as signals that are reasonably distinct from those of good utterances, which are supposed to contain intelligible human speech. This motivates us to train a binary classifier using the audio data to directly differentiate uninteresting utterances from the good ones.

In addition to the audio data, we also have access to a variety of metadata, such as the length of the audio, the volume level, etc. Moreover, by passing the acoustic features through the ASR model, we obtain additional features such as the recognition confidence. Therefore, we propose to combine both acoustic features extracted from raw audio and selected metadata variables to build our classification model. Given the two types of features, we train a neural network model on the acoustic features, and train an XGBoost model [12] on the metadata, separately. Finally, we take the probabilities produced for each utterance by both the neural network and XGBoost, and train a logistic regression classifier on these probabilities, in order to combine the two models together. We term the final ensemble model as the CAM detector, reflecting the fact that it operates on a **C**ombination of **A**udio and **M**etadata features.

For the neural network, we explore a variety of different architectures. In particular, one architecture we experiment with is the baseline model in [1], which combines a CNN and a GRU [13]. As mentioned in the related work section, [1] deals with media speech detection, which we identify as one reason that renders an utterance uninteresting. Therefore, we deem the work in [1] as relevant to our task. However, their main models work with multi-channel audio data, which are not available in our case. As a result, we decide to experiment with their baseline model, which operates on mono-channel data and is the current state-of-the-art architecture on such data.

Below, we discuss the feature extraction process and our models in more detail.

3.1. Feature extraction

3.1.1. Acoustic features

For the audio data, we experiment with two types of features. The first type is mel fequency cepstral coefficents (MFCCs) [14], which are widely used in automatic speech and speaker recognition. For the second type, we follow [1] and use log filterbank energies (LFBEs). The process to obtain both types of features is described below.

For each utterance, we first segment the audio into small frames using a 25 ms window and a 10 ms step. Then we apply short-time Fourier transformation to each frame and obtain its power spectrum. The energies of nearby frequencies are then binned together using overlapping mel-scale filterbanks. Taking the log of the resulting energies in different filterbanks produces the LFBE features. The MFCCs can then be obtained by applying a discrete cosine transformation to LFBEs and keeping only the lower coefficients. In the end, the audio data of each utterance is transformed into a matrix of dimension (number of frames, number of features), where the second dimension is 13 for MFCCs and 64 for LFBEs. For the frame dimension, we zero-pad all the instances to have 498 frames, and those with more are truncated.

In addition, we also experiment with another way of processing the frame dimension, which we call voting. Figure 1 shows an illustration of the procedure. In this setting, each instance is split into smaller instances using a window of 100 frames which advances 50 frames at a time. The label of the original instance is copied over to these sub-instances, which are then used to train the model. For prediction, the scores of all the sub-instances that belong to the same utterance are aggregated by taking either the mean or the maximum.

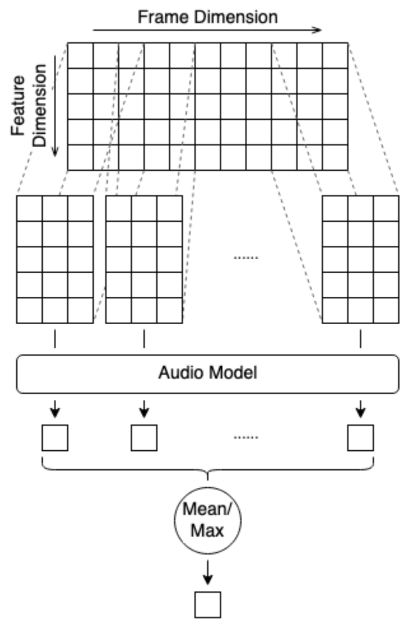

Figure 1: *Illustration of how one utterance is split up in the voting procedure. In the figure, the original instance has dimension 11 × 5. A sliding window of size 3 and stride 2 is used to split up the original instance into 5 smaller instances of shape 3 × 5. For training, the label of the original instance is copied over to the smaller instances, which are used to train the model. For inferencing, the scores of the smaller instances are aggregated through mean or max pooling to produce the score of the original instance.*

3.1.2. Metadata features

Below is a list of the 11 metadata variables that we choose to train our model with. All the variables have numerical values which are nonnegative. The values of some variables could be missing for some utterances, and we use -1 to denote such cases.

1) Audio duration in milliseconds
2) Current audio player volume of the voice assistant

3) Current notification volume of the voice assistant
4) Audio noise energy level
5) Audio voice energy level
6) Background audio strength
7) Foreground audio strength
8) Confidence score of the presence of non-verbal speech
9) Confidence score of the ASR hypothesis
10) Confidence score of the presence of a wake word (the phrase by which the voice assistant is activated)
11) Threshold to activate the assistant given the wake word confidence score

3.2. Models

As mentioned in the beginning of the section, we first train a neural network, which takes in acoustic features only. Then an XGBoost model is trained using the 11 metadata variables. The performance of the two models can be viewed as a reflection of the predicting power of the two different kinds of features, respectively. Finally, the two models are combined in the CAM detector, thus taking advantage of both types of features.

The various architectures that we explore for the neural network model are discussed below. As for XGBoost, we use the XGBClassifier class as provided in the official Python implementation.[1] For the CAM detector, we first gather the probabilities produced by both the neural network and the XGBoost model for each utterance, and then train a logistic regression classifier using these probabilities as features.

For the neural network model, we first use a sequence-to-vector encoder to transform an input audio into a continuous low-dimensional representation. The result is then passed through an 1-layer feedforward network, which has 64 hidden units, uses ReLU activation, and has a dropout rate of 0.2. This is followed by a linear output layer to obtain a score for classification. For the encoder, we experiment with the following architectures:

Feedforward neural network (FF)

This encoder serves as a simple baseline. It first averages the features across all frames, and then passes the result through an 1-layer feedforward network, which has 128 hidden units, uses ReLU activation, and has a dropout rate of 0.2.

LSTM

This encoder consists of a single layer of bi-directional LSTM [15], where the hidden dimension in each direction is 32.

NGram CNN

This is the n-gram CNN model that is typically used for text classification [16]. In our case, we use filters that span 4, 8, 16, and 32 frames, and maintain 16 filters of each type. The results from each filter are max-pooled and then concatenated together as output.

CNN

This CNN is adapted from the one used in the baseline of [1]. It first applies a 2D convolutional layer with a 5×5 kernel and 8 output channels, which is followed by ReLU activation and a 2D max-pooling layer with a 1×3 kernel. Then there is a second 2D convolutional layer with a 5×5 kernel and 32 output channels, followed by ReLU and 2D max-pooling with a 1×3 kernel. The channel and feature dimensions of the result is then flattened, and averaged across the frame dimension. Finally, an 1-layer feedforward network is applied, where the hidden dimension is 64, the activation is ReLU, and the dropout rate is 0.1.

CNN + GRU

This is the baseline model from [1]. The first part of the encoder is the same as the CNN model above up until the flatten operation. Afterwards, the result is not averaged across the frame direction, but rather, an 1-layer uni-directional GRU is applied, where the hidden dimension is 64. The result is then passed through a feedforward network which is the same as the one used in the CNN model.

4. Experiment Results

4.1. Dataset and model hyper-parameters

The dataset we have consists of around 43K utterances, out of which around 4K are considered uninteresting. We randomly shuffled the dataset and split it 60/20/20 into train/valid/test sets.

The audio of the utterances all has a single channel and has a sample rate of 16000 samples/sec. The sample size is 2 byte, or 16 bits, and thus the raw audio is represented as an array of 16-bit integers. As for the acoustic features, for MFCCs, we follow the convention and use 26 filterbanks that span frequencies from 0 kHz to 8 kHz, and keep only the first 13 MFCCs, with the first coefficient replaced by the log of the total frame energy. For LFBEs, we follow [1] and use 64 filterbanks that span 0.1 - 7.2 kHz.

All of the neural network models are trained using Adam [17] with default parameters. The batch size is 128 and training stops until the F1 score on the validation set stops increasing for 10 consecutive epochs. The XGBoost model is trained using its default parameters as defined in the XGBClassifier class, and the logistic regression we use is from Scikit-Learn [2], which we train using the default parameters except for max iteration, which is set to 5000.

4.2. Results and analysis

Table 1 shows the results of the various neural networks that we experiment with. The numbers are relative to the performance of the FF model (leftmost column). All the models are trained using MFCC features except for CNN + GRU, which we also train using LFBE features, and with the voting schemes as described in Section 3.1.1.

Looking at the columns under MFCC only, we can see that the CNN + GRU model gives the best performance, except for precision, on which the FF model achieves the best. In particular, we calculate the average precision score [3] for each model, which is a measure that summarizes the precision-recall curve. It can be observed that the CNN + GRU beats all other models in this metric on both the validation set and the test set, showing that this model indeed has the best performance overall.

Next, comparing the performance of CNN + GRU under MFCC and that under LFBE, we can clearly see that using the LFBE features produces a big improvement over using MFCCs. The last two columns in the table are the CNN + GRU model trained using the voting schemes that are described in Section 3.1.1. Both of these appear to have a really low recall, with

[1]`https://xgboost.readthedocs.io/en/latest/`

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

[3]`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html`

		MFCC					LFBE		
								Voting	
		FF	LSTM	NGram CNN	CNN	CNN + GRU	CNN + GRU	CNN + GRU (Mean)	CNN + GRU (Max)
Train	Acc.	1.00	1.02	1.04	1.01	1.02	1.03	0.99	1.00
	Prec.	1.00	0.97	1.03	0.91	0.99	1.06	1.11	0.90
	Rec.	1.00	1.75	2.63	1.72	2.02	2.07	0.56	1.37
	F1	1.00	1.42	1.81	1.37	1.56	1.63	0.65	1.19
	Avg Prec.	1.00	1.27	1.63	1.22	1.40	1.50	0.88	1.12
Valid	Acc.	1.00	0.99	0.98	1.00	1.00	**1.01**	1.00	0.98
	Prec.	1.00	0.78	0.69	0.84	0.82	0.91	**1.17**	0.64
	Rec.	1.00	1.28	1.67	1.60	1.76	**1.77**	0.55	1.01
	F1	1.00	1.08	1.19	1.27	1.33	**1.40**	0.65	0.87
	Avg Prec.	1.00	0.99	1.09	1.21	1.30	**1.38**	0.95	0.85
Test	Acc.	1.00	1.00	0.99	1.00	1.00	**1.01**	1.00	0.99
	Prec.	1.00	0.83	0.77	0.90	0.88	0.95	**1.27**	0.73
	Rec.	1.00	1.52	1.93	1.79	1.94	**2.00**	0.60	1.06
	F1	1.00	1.24	1.36	1.40	1.46	**1.53**	0.71	0.94
	Avg Prec.	1.00	1.05	1.19	1.28	1.37	**1.42**	1.06	0.95

Table 1: *Results of various neural network architectures, using either MFCCs or LFBEs as features, with and without using a voting scheme as described in Section 3.1.1. The numbers are relative to the performance of the FF model (leftmost column). For the validation and test sets, best results are in bold, and the second best are underlined.*

		Audio Model	Metadata Model	CAM
		CNN + GRU (LFBE)	XGBoost	Logistic Regression
Train	Acc.	1.03	1.04	1.05
	Prec.	1.06	1.15	1.17
	Rec.	2.07	2.38	2.56
	F1	1.63	1.82	1.91
	Avg Prec.	1.50	1.65	1.69
Valid	Acc.	1.01	1.01	**1.02**
	Prec.	0.91	0.96	**1.01**
	Rec.	1.77	1.82	**2.07**
	F1	1.40	1.45	**1.59**
	Avg Prec.	1.38	1.48	**1.57**
Test	Acc.	1.01	1.02	**1.03**
	Prec.	0.95	1.02	**1.08**
	Rec.	2.00	2.10	**2.37**
	F1	1.53	1.62	**1.78**
	Avg Prec.	1.42	1.56	**1.65**

Table 2: *Results of the audio model (CNN + GRU trained using LFBE), the metadata model (XGBoost), and the CAM detector (logistic regression). The numbers are relative to the performance of the FF model (leftmost column in Table 1). For the validation and test sets, best results are in bold, and the second best are underlined.*

the mean setting being much worse than the max setting. In terms of precision, the max setting still performs poorly, but the mean setting is actually able to achieve a really high precision. Having a high precision and a low recall as such indicates that the model probably tends to predict very few positives, which suggests that this voting scheme might actually have made the model more prone to the class imbalance problem.

Furthermore, from the average precision scores in the last four columns in Table 1, we again confirm that CNN + GRU trained using LFBEs has the best overall performance, while the two models trained using the voting schemes produce very poor results.

Moving on to the metadata model and the CAM detector, their results are shown in the Table 2, where the results of the CNN + GRU model trained using LFBEs are also included for comparison. The numbers are again relative to the performance of the FF model (leftmost column in Table 1). We can see that the metadata model is slightly better than the audio model, which shows that the metadata indeed provide very useful information in differentiating between good utterances and uninteresting ones. Meanwhile, by combining the neural network and XGBoost together through a logistic regression model, the CAM detector is able to achieve a significant boost in performance, which beats the results of using either the audio model or the metadata model alone by a large margin. For example, CAM detector's F1 score on the test set is 1.78, which translates to a 16.3% increase over the audio model's 1.53, and a 9.9.% increase over the metadata model's 1.62. This demonstrates that the audio data and the metadata provide distinct signals for identifying uninteresting utterances, and we can harvest the most gains by complementing them with each other.

5. Conclusions and Future Work

In this paper, we propose the CAM detector, which combines both acoustic features and selected metadata variables to filter out uninteresting utterances. Our experimental results show that the CAM detector achieves a dramatic performance boost when compared to using either the audio model or the metadata model alone. Having such a system allows us to identify uninteresting utterances at an early stage in the processing pipeline, thus achieving our aforementioned goals of saving system resources and increasing efficiency. In the future, we will explore using more advanced network architectures in the audio model, as well as using more variables from the metadata to boost the performance. We will also experiment with data augmentation and semi-supervised methods to enrich our training samples.

A. References

[1] C. Papayiannis, J. Amoh, V. Rozgic, S. Sundaram, and C. Wang, "Detecting media sound presence in acoustic scenes," in *Conference of the International Speech Communication Association*, 2018.

[2] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," in *IEEE Signal Processing Magazine*, 2015.

[3] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of lstm and cnn," in *Detection and Classification of Acoustic Scenes and Events*, 2016.

[4] S. Adavanne, P. Pertilä, and T. Virtanen, "Sound event detection using spatial features and convolutional recurrent neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[5] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[6] Z. Ren, Q. Kong, K. Qian, M. D. Plumbley, B. Schuller *et al.*, "Attention-based convolutional neural networks for acoustic scene classification," in *Detection and Classification of Acoustic Scenes and Events*, 2018.

[7] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," in *EURASIP Journal on Audio, Speech, and Music Processing*, 2015.

[8] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," in *Conference of the International Speech Communication Association*, 2016.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2014.

[10] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Feature learning with matrix factorization applied to acoustic scene classification," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2017.

[11] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *European Signal Processing Conference*, 2010.

[12] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[13] K. Cho, B. van Merrienboer, aglar Gülehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[14] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken se," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1980.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," in *Neural Computation*, 1997.

[16] Y. Kim, "Convolutional neural networks for sentence classification," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.