Speech Detection with Daimensions

Accurate speech detection is desirable in many real world systems. For instance, transmitting silence packets wastes compute and bandwidth, thus it is valuable to detect these non-speech instances first. In audio compression, speech has unique properties that allow for different and more effective compression schemes, so detecting speech from other audio signals is also important. The following outlines an approach to speech/non-speech/music detection made in collaboration with Brainome and their Daimensions technology.

A common approach to audio analysis consists of the following three parts. First, the audio is windowed, since the relevant information is likely contained in a brief frame of time. Second, features are extracted from each window to reduce noise and dimension. Third, the extracted features are used as inputs to a predictor.

The choice of window is very task dependent. For speech detection, a short window of 50 milliseconds can be used. A very successful method for audio feature extraction is to take the mel frequency cepstral coefficients (MFCCs) of the windowed signal [10, 7, 8]. Computing the MFCCs is a multi-step process that approximates a compressed, decorrelated representation of the signal's power spectrum on the mel scale [9].

Typically, the MFCCs would be determined for each window of the original signal. We take a slightly different yet beneficial approach. In addition to determining the MFCCs for a given window, which we will call the parent window, we choose a series of sub-windows which span the parent window and compute the MFCCs for each sub-window. The MFCCs from the parent window together with the MFCCs from each sub-window are then used as extracted features. The motivation is the signal may contain correlations at different time scales.

Once the audio features have been extracted, we build a predictor using Brainome's Daimensions product. Daimensions uses methods originating from information theory to architect and train a model. Information theory has long been

a useful tool in machine learning [6, 11]. In particular, Daimensions chooses a model by measuring the propensity of the model to generalize given the data, since it has been shown that even random points can be memorized by a large model [2]. The benefits of this approach go beyond generalization, since by determining the correct size and architecture for the model prior to training, the time cost of manually tuning-then-training the model is saved.

This end-to-end framework has been tested on a data set composed of speech from the AMI corpus, non-speech from the DCASE task 5 data set, and music from the FMA data set [4, 3, 5]. Speech instances are sourced from the AMI data set using the manually transcribed, word-level start and end times. Non-speech instances are composed of all instances in the DCASE data set which are not classified as "social interaction" and "watching television" since these classes regularly include speech.

To produce a predictor requires two steps. The first step is to transform the raw audio into a CSV of MFCC features using code found at this github [1]. This program uses the popular Librosa Python library [1]. The particular command to execute is

```
python make_data.py -gb 2. --outsize 0.45
```

This truncates the data creation process as the whole data set is very large. With the directory structure described on the github's README, a CSV will be created at `root/datadir/bar/combined_train.csv`. After this, Daimensions is called on the CSV. This can be done using

```
btc -vv -f NN -o output.py /path/to/csv
```

It will measure the data complexity to determine an appropriate model, then it may perform whitening or dimensionality reduction on the data as needed, and finally it will train and output the predictor along with summary statistics validating the predictor's quality. On the previously described data set, the resulting predictor was small, had accuracy above 95% and a high generalization ratio. It can be found on the github in the `predictors` directory.

The predictor (and any data transformation) is returned as a method implemented in `output.py` using Numpy and otherwise built-in Python functions. It is common for the predictor to be small due to Daimension's data measuring process. As such, deploying a Daimensions predictor on another device or predicting on novel inputs is very easy.

[1] https://github.com/Zachary-Stone-Berkeley/Speech. This code works generally on any data set that adheres to the guidelines found in the github README.

Bibliography

[1] Brian McFee et al. "librosa: Audio and Music Signal Analysis in Python". In: (2015). URL: http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf.

[2] Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: (2016). URL: https://arxiv.org/abs/1611.03530.

[3] Gert Dekkers et al. "DCASE 2018 Challenge - Task 5: Monitoring of domestic activities based on multi-channel acoustics". In: (2018). URL: https://arxiv.org/abs/1807.11246.

[4] Jean Carletta et al. "The AMI Meeting Corpus: A Pre-Announcement". In: (2006). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.380.3306&rep=rep1&type=pdf.

[5] Michaël Defferrard et al. "FMA: A Dataset For Music Analysis". In: (2016). URL: https://arxiv.org/abs/1612.01840.

[6] Naftali Tishby et al. "The information bottleneck method". In: (2000). URL: https://arxiv.org/abs/physics/0004057.

[7] Shekhar Nayak et al. "An investigation into instantaneous frequency estimation methods for improved speech recognition features". In: (2017). URL: https://ieeexplore.ieee.org/abstract/document/8308665.

[8] Jaeyoung Choi and Gerald Friedland. *Multimodal Location Estimation of Videos and Images*. 2014.

[9] S.B. Davis and P. Mermelstein. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1980). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.380.3306&rep=rep1&type=pdf.

[10] Meinard Müller. *Information Retrieval for Music and Motion*. 2007, p. 65.

[11] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. 1992.