The Journal of Open Source Software

Spafe: Simplified python audio features extraction

Ayoub Malek [1]

1 Yoummday GmbH

DOI: 10.21105/joss.04739

Software

- Review
- Repository
- Archive

Editor: Fabian-Robert Stöter

Reviewers:

- @hadware
- @hbredin

Submitted: 14 July 2022
Published: 27 January 2023

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

This paper describes version 0.3.1 of spafe: a python package for audio features extraction based on the Numpy (Harris et al., 2020) and Scipy (Pauli Virtanen & Vázquez-Baeza, 2019) libraries. Spafe implements various features extraction techniques that can be used to solve a wide variety of recognition and classification tasks (speaker verification, spoken emotion recognition, spoken language identification etc.). The paper provides a brief overview of the library's structure, theory and functionalities.

Statement of need

In speech processing, features extraction is essentially the estimation of a parametric representation of an input signal. This is a key step in any audio based modeling and recognition process (e.g. speech recognition, sound classification, speaker authentication etc.). There are several speech features to extract, such as the Linear Frequency Cepstral Coefficients (LFCC), Mel Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding (LPC), and Constant-Q Cepstral Coefficients (CQCC) etc. Each type of features has its own advantages and drawbacks (e.g. noise robustness, complexity, inter-components correlation etc.) that can directly affect the researched topic. Unfortunately, existing libraries for extracting these features (e.g. librosa (McFee et al., 2015), python_speech_features (Lyons et al., 2020), SpeechPy (Torfi, 2018) and Bob (A. Anjos & Marcel, 2017)) are limited and mostly focus on one extraction technique (e.g. MFCC), thus it is hard to find reliable implementations of other features extraction algorithms. Consequently, this slows down the research and hinders the possibility of exploring and comparing these different approaches. Hence, the need for **spafe**, a straightforward solution that unites all these different techniques in one python package.

Introduction

The philosophy of spafe is keeping it simple, flexible and efficient in order to reach a wide range of developers and researchers. Hence, spafe is written in python 3 and only depends on Numpy (Harris et al., 2020) and Scipy (Pauli Virtanen & Vázquez-Baeza, 2019). The library is heavily documented with the help of Sphinx and tested using Pytest. Spafe supports **mono signals processing** and has been tested with different sampling rates (e.g. 8kHz, 16Khz, 44.1kHz, 48kHz etc.).

Scripts in spafe are divided into four major groups (see Figure 1):

- **fbanks** filter banks implementations.
- **features** features extraction implementations.
- **frequencies** frequencies based features extraction implementations.
- **utils** helper functions for pre- & post-processing and visualization etc.

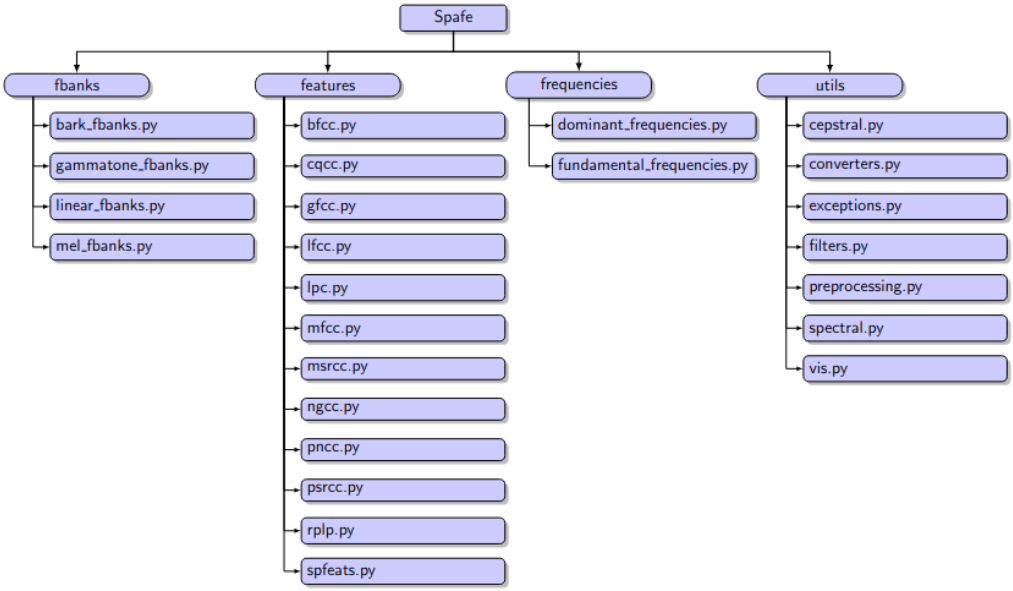

Figure 1: Structure of spafe.

Implementation and theory

Filter banks (`spafe/fbanks`)

A filter bank is defined as an array of band pass filters that splits the input signal into a set of analysis signals, each one carrying a single frequency sub-band of the original signal (Penedo et al., 2019; Sarangi et al., 2020). Each band pass filter is centered at a different frequency, called center frequency. The center frequencies are evenly spaced over a specificied scaled frequencies range(e.g. bark scale, erb scale, mel scale etc.). The bandwidths of the filters increase with the frequency, in order to duplicate the human hearing properties, which are characterized by a decreasing sensitivity at higher frequencies. Within this context, spafe provides implementations for the following filter banks: **Bark filter banks**, **Gammatone filter banks**, **Linear filter banks** and the **Mel filter banks**.

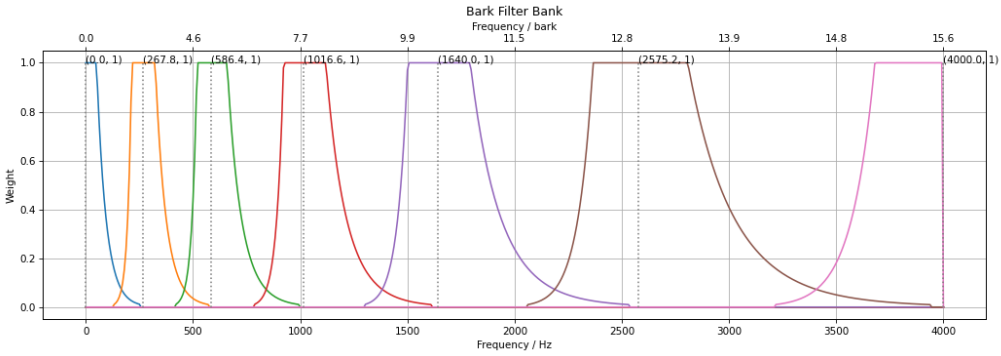

Figure 2: Bark filter banks computed and visualized using spafe

Features (`spafe/features`)

In an attempt to cover most audio features, spafe provides various frequency and cepstral domain features extraction algorithms, both filter bank-based and auto-regression-based. The

following is a list of the available features extraction routines in the spafe python package:

- Bark Frequency Cepstral Coefficients	BFCC
- Constant Q-transform Cepstral Coefficients	CQCC
- Gammatone Frequency Cepstral Coefficients	GFCC
- Linear Frequency Cepstral Coefficients	LFCC
- Linear Prediction Cepstral Coefficients	LPCC
- Mel Frequency Cepstral Coefficients	MFCC
- Inverse Mel Frequency Cepstral Coefficients	IMFCC
- Magnitude based Spectral Root Cepstral Coefficients	MSRCC
- Normalized Gammachirp Cepstral Coefficients	NGCC
- Power-Normalized Cepstral Coefficients	PNCC
- Phase based Spectral Root Cepstral Coefficients	PSRCC
- Perceptual Linear Prediction Coefficents	PLP
- Rasta Perceptual Linear Prediction Coefficents	RPLP

The following figure provides a summary of the included features extraction algorithms and their detailed steps:

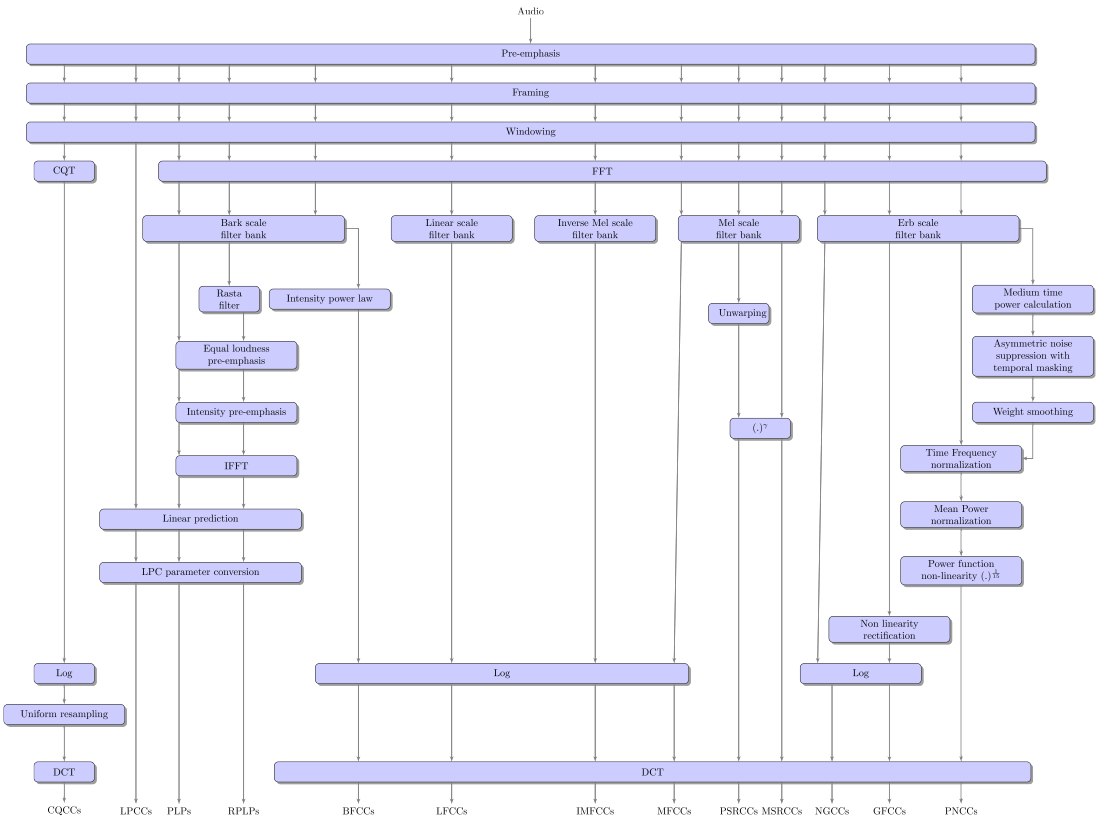

Figure 3: Features extraction algorithms in spafe

In addition to the previously mentioned features, spafe allows for computing the following spectrograms: **Bark spectrogram**, **Cqt spectrogram**, **Erb spectrogram** and **Mel spectrogram**.

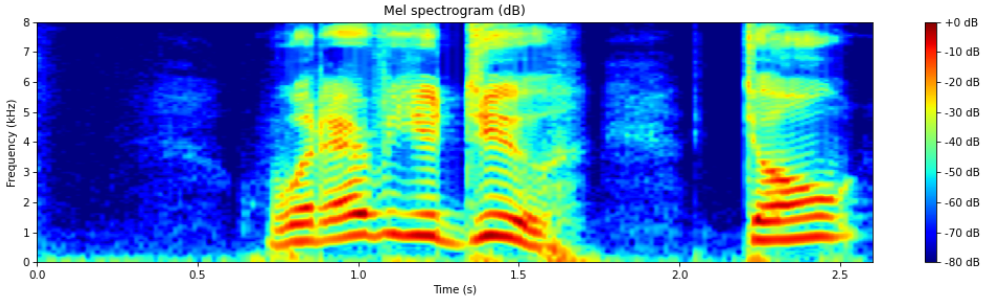

Figure 4: Mel spectrogram computed and visualized using spafe

Frequencies (`spafe/frequencies`)

The frequencies modules in spafe focus specifically on the computation of dominant and fundamental frequencies. A dominant frequency is per definition the frequency carrying the maximum energy among all frequencies of the spectrum (Telgarsky, 2013), whereas the fundamental frequency (often noted as F_0) is defined as the inverse of the period of a periodic signal (Cheveigné & Kawahara, 2002).

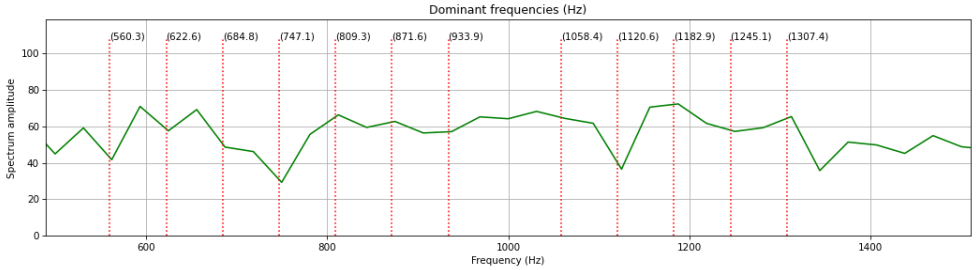

Figure 5: Dominant frequencies computed and visualized using spafe

Utils (`spafe/utils`)

The utils scripts, handle most of the input signal pre-processing steps including pre-emphasis, framing and windowing. They also include all the conversion computations needed to convert Hertz frequencies to other frequency scales. On top of that, all feature post-processing routines are in this group. This includes normalization, liftering, deltas computation and visualization.

Conclusion

This paper introduced spafe, a python package for audio feature extractions. Spafe provides a unified solution for audio features extraction, that can help simplify and accelerate the research of various audio-based recognition experiments.

References

A. Anjos, T. de F. P., M. Günther, & Marcel, S. (2017, August). Continuously reproducing toolchains in pattern recognition and machine learning experiments. *International Conference on Machine Learning (ICML)*. http://publications.idiap.ch/downloads/papers/2017/Anjos_ICML2017-2_2017.pdf

Cheveigné, A. de, & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. In *The Journal of the Acoustical Society of America* (No. 4; Vol. 111, pp. 1917–1930). Acoustical Society of America (ASA). https://doi.org/10.1121/1.1458024

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lyons, J., Wang, D. Y.-B., Gianluca, Shteingart, H., Mavrinac, E., Gaurkar, Y., Watcharawisetkul, W., Birch, S., Zhihe, L., Hölzl, J., Lesinskis, J., Almér, H., Lord, C., & Stark, A. (2020). *Jameslyons/python_speech_features: Release v0.6.1*. Zenodo. https://doi.org/10.5281/ZENODO.3607820

McFee, Brian, Raffel, Colin, Liang, Dawen, Ellis, Daniel P.W., McVicar, Matt, Battenberg, Eric, & Nieto, Oriol. (2015). Librosa: Audio and Music Signal Analysis in Python. In Kathryn Huff & James Bergstra (Eds.), *Proceedings of the 14th Python in Science Conference* (pp. 18–24). https://doi.org/10.25080/Majora-7b98e3ed-003

Pauli Virtanen, T. E. O., Ralf Gommers, & Vázquez-Baeza, Y. (2019). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 1–12. https://doi.org/10.1038/s41592-019-0686-2

Penedo, S. R. M., Netto, M. L., & Justo, J. F. (2019). Designing digital filter banks using wavelets. In *EURASIP Journal on Advances in Signal Processing* (No. 1; Vol. 2019). Springer Science; Business Media LLC. https://doi.org/10.1186/s13634-019-0632-6

Sarangi, S., Sahidullah, M., & Saha, G. (2020). Optimization of data-driven filterbank for automatic speaker verification. In *Digital Signal Processing* (Vol. 104, p. 102795). Elsevier BV. https://doi.org/10.1016/j.dsp.2020.102795

Telgarsky, R. (2013). Dominant frequency extraction. *CoRR*, *abs/1306.0103*. http://arxiv.org/abs/1306.0103

Torfi, A. (2018). SpeechPy - a library for speech processing and recognition. *Journal of Open Source Software*, *3*(27), 749. https://doi.org/10.21105/joss.00749