AWtoolbox: Characterizing Audio Information Using Audio Words

Chin-Chia Michael Yeh, Ping-Keng Jao, and Yi-Hsuan Yang
Research Center for IT Innovation, Academia Sinica, Taiwan
{mcyeh, nafraw, yang}@citi.sinica.edu.tw

ABSTRACT

This paper presents the AWtoolbox, an open-source software designed for extracting the audio word (AW) representation of audio signals. The toolbox comes with a graphical user interface that helps a user design custom AW extraction pipelines and various algorithms for feature encoding, dictionary learning, result rectification, pooling, normalization and others. This paper also reports a benchmark comparing eight AW representations computed by the toolbox against state-of-the-art low-level and mid-level timbre, rhythmic and tonal descriptors of music and sound. The evaluation result shows that sparse coding (SC) based AW representation leads to very competitive performances across the three tested sound and music classification tasks. AWtoolbox is available for download at `http://mac.citi.sinica.edu.tw/awtoolbox`.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Multimedia Information Systems, Sound and Music Computing

Keywords

Audio feature; audio word; vector quantization; sparse coding; sound classification; music auto-tagging

1. INTRODUCTION

Representing audio information in a symbolic and text-like fashion analogous to the bag-of-words feature prevalent in text classification is not a new topic. The classic vector quantization approach represents a series of audio feature vectors extracted from an audio signal by the occurrence of codewords in a pre-built dictionary (codebook), leading to the so-called audio word (AW) representation for the entire signal [9, 12]. Comparing to conventional audio features, an AW representation is characteristic of its ability of symbolizing any local audio event as a codeword, its flexibility of using an arbitrary large number of codewords learned from a corpus of audio data in an unsupervised fashion, and its light dependence on domain knowledge for feature design.

MM'14, November 3–7, 2014, Orlando, Florida, USA.

ACM 978-1-4503-3063-3/14/11 ...$15.00.
http://dx.doi.org/10.1145/2647868.2654989 .

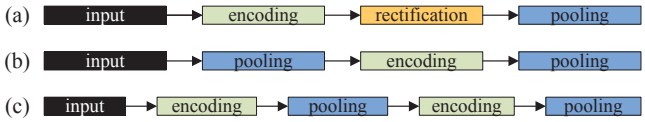

Figure 1: Three examples of AW extraction pipeline.

Therefore, it has been considered an powerful alternative to conventional hand-crafted audio features.

However, due to the advancement of AW encoding and dictionary learning algorithms [2, 11, 13, 17, 19], it is not easy to keep track of different proposals of AW extraction algorithms and make comprehensive comparisons. For example, sparse coding (SC) has received considerable attention in recent years and its variants have been used extensively in various audio-related research [4, 14, 15]. Post-processing methods such as encoding result rectification and normalization [3, 7] have also been claimed important. Because of the lack of a standardized implementation of the related algorithms under a unified framework, it is difficult to compare the results reported in different works and gain insights.

In light of this, we present in this paper the AWtoolbox, an open-source graphic user interface application intended to facilitate the implementation and development of various AW extraction pipelines. We also report a benchmark comparing the performance of various AW representations calculated by the toolbox and conventional audio features for three different sound and music classification tasks.

2. AWTOOLBOX OVERVIEW

We define five atomic functional layers of AW extraction: `input`, `encoding`, `rectification`, `pooling` and `other`, whose details are presented later. As Figure 1 illustrates, different AW representations can be obtained by not only using different algorithms for each layer, but also cascading the functional layers in different ways. The same layer can be applied multiple times, using not necessarily the same algorithm each time. It is this versatility of the AW representation that makes it important to allow the users to define the number and order of these layers on their own. In a nutshell, with the proposed toolbox, extracting a custom AW representation can be done with the following steps:

- **Designing**: Users can graphically design the process by creating and arranging various kinds of layers for generating the desired AW representation. For visualization purpose, layers are color coded based on their types. For instance, the `input` layer is colored black and the `pooling` layer is colored light blue.
- **Dictionary learning**: Users can either provide a previous built dictionary or prepare a corpus for con-

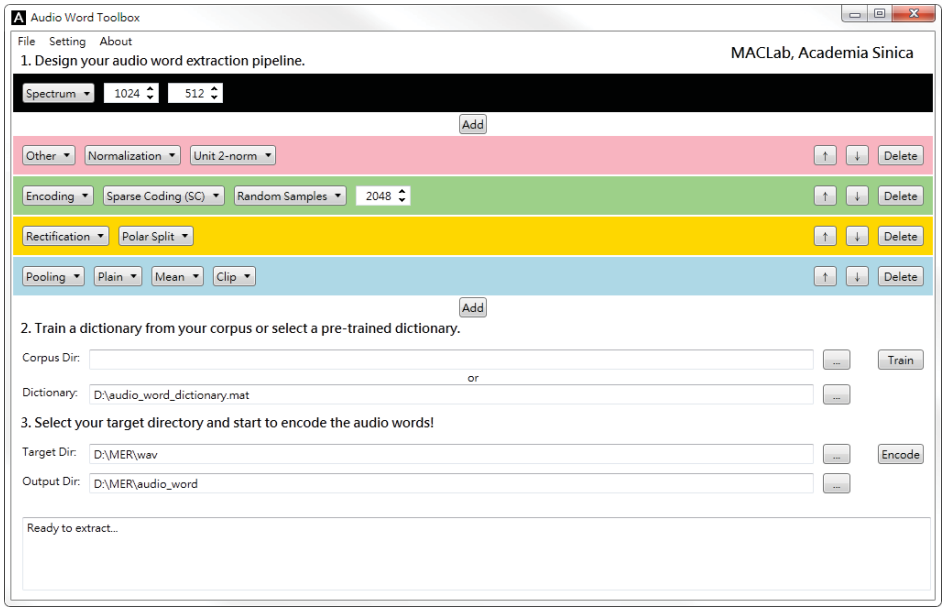

Figure 2: An screenshot of the graphical user interface of AWtoolbox.

structing the dictionary. The dictionary and the corresponding user-specified design can be saved for later use.

- **Audio word encoding**: When the desired dictionary is trained or selected, all the audio clips under the input directory will be encoded to generate the AW representation once the "Encode" button is pressed.

Figure 2 shows the graphical user interface of AWtoolbox. Currently, AWtoolbox is only available for the Windows platform. It is written in C#, with .NET assembly built from MATLAB codes for the underlying algorithms. The C# and MATLAB source codes are distributed under GPLv3 license. Please visit AWtoolbox's website for example workflow on AW extraction.

The `input` layer is the first layer in any AW extraction pipeline, transforming an input audio stream into a series of t frame-level feature vectors that are further processed in subsequent layers. Low-level feature representations applicable here include log power spectrum (SPEC), Mel-frequency cepstral coefficients (MFCC), sonogram, and chromagram, among others, extracted with the frame rate and hop size specified by a user. While MFCC-based AWs have been found effective in both audio and video classification problems [9, 12], it has been reported [15] that sparse coding works better with primitive features (e.g. SPEC) instead of sophisticated features because primitive features preserve more details of the raw signal.

The `encoding` layer converts an input vector (e.g. MFCC) $\mathbf{x} \in \mathbb{R}^m$ into the encoding result $\boldsymbol{\alpha} \in \mathbb{R}^k$ using a dictionary $\mathbf{D} \in \mathbb{R}^{m \times k}$ and an encoding method, where k denotes the number of codewords in $\mathbf{D}$. When there are t input vectors from an audio clip, we encode each vector individually. This involves the specification of an encoding algorithm and a dictionary learning algorithm. If desired, `encoding` can be applied multiple times, as Figure 1(c) depicts, to mimic the structure of deep neural nets [4,13]. Specifically, we consider the following four state-of-the-art encoding algorithms:

- **Vector quantization (VQ)** represents $\mathbf{x}$ by a one-hot binary vector $\boldsymbol{\alpha}$ according to the nearest codeword $\mathbf{d}_j \in \mathbb{R}^m$ in $\mathbf{D}$. Namely, only an α_j is 1 and the rest of $\boldsymbol{\alpha}$ are 0, where $j = \text{argmin}_p\ z_p$ and $z_p = \|\mathbf{x} - \mathbf{d}_p\|_2^2$.
- **Triangle coding (TC)**, a 'soft' variant of VQ [12], obtains a real-valued $\boldsymbol{\alpha}$ by $\alpha_j = \max\{0, \mu(\mathbf{z}) - z_j\}, \forall j$, where $\mu(\mathbf{z}) = \frac{1}{k}\sum_{p=1}^{k} z_p$ is the mean of these distances.
- **Sparse coding (SC)** represents the input signal by a sparse combination of the dictionary codewords by solving the following LASSO problem [2],

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\text{argmin}} \frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1\,, \qquad (1)$$

  where λ controls the balance between the reconstruction error $\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2$ and the sparsity $\|\boldsymbol{\alpha}\|_1 = \sum|\alpha_j|$, which is a convex relaxation of the l_0 norm $\|\boldsymbol{\alpha}\|_0 = \sum|\alpha_j|^0$. A recommended value for λ is $1/\sqrt{\min(m,k)}$ [11]. For the case of $k \gg m$, it has been shown that SC outperforms VQ for audio classification problems [15].
- **Modified LASSO screening coding (SCS)** is a variant of SC with much lower computational cost due to a theoretically-justified mechanism to filter out codewords not useful for reconstructing the input signal before solving Eq. 1 [17]. We adopt an algorithm tailored for audio signals proposed in [8] and employ clip-level rather than frame-level screening for better efficiency in time and memory usage. With SCS, we can afford using larger k for the dictionary.

As for dictionary learning, the following three widely-used, unsupervised algorithms are considered:

- **k-means** generates a dictionary by using each cluster center as a codeword after applying k-means clustering to the training corpus. This algorithm is usually used for VQ-based representation [9, 12].
- **Online dictionary learning (ODL)** learns a dictionary by optimizing the following equation using stochastic gradient descent [11],

$$\mathbf{D}^* = \underset{\mathbf{D}}{\text{argmin}} \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{2}\|\mathbf{x}^{(n)} - \mathbf{D}\boldsymbol{\alpha}^{(n)}\|_2^2 + \lambda\|\boldsymbol{\alpha}^{(n)}\|_1\right), \qquad (2)$$

where N denotes the number of low-level feature vectors in the training corpus and n indexes the training instances. Variants of Eq. 2 that consider other cost functions such as non-negativity, group sparisty and structure sparsity have also been proposed [2].

- **Random exemplar extraction (REE)** randomly extracts k examples from the training corpus and directly uses the extracted examples as codewords for the dictionary. Therefore, it bypasses the computational cost involved in clustering or solving Eq. 2. It has been found that using such a random dictionary is effective when the dictionary size k is large [8].

The `rectification` layer applies rectifying non-linearity to the encoding result for improving representation power [3].

- **Absolute value** simply applies the absolute value function to all the elements of the input to this layer.
- **Polarity splitting** splits the positive and negative elements of the input data into separate ones and concatenates them after changing the sign of the negative ones [3]. For example, as Figure 1(a) shows, when the input is the time-varying encoding result $\mathbf{A} \in \mathbb{R}^{k \times t}$, the output of polarity splitting would be $\hat{\mathbf{A}} \in \mathbb{R}^{2k \times t}$, $\hat{\mathbf{A}} = [\max\{0, \mathbf{A}\}^{\mathrm{T}},\ \max\{0, -\mathbf{A}\}^{\mathrm{T}}]^{\mathrm{T}}$.

The `pooling` layer summarizes a time-varying vector sequence by aggregation operators such as taking the mean or maximum or by other advanced multi-scale pooling techniques such as temporal pyramid pooling [7]. Pooling can be done either in the clip-level or in the segment-level (a segment is a subset of a clip consisting of multiple consecutive frames), and after or before the `encoding` layer. For example, in Figure 1(a) the `pooling` layer combines the frame-level encoding results $\mathbf{A} \in \mathbb{R}^{k \times t}$ to form a k-dimensional clip-level feature vector (e.g. $\frac{1}{t}\sum \boldsymbol{\alpha}_i$ for the case of mean pooling) that can be used for classification. Figure 1(b) shows another example that pools the frame-level low-level features into segment-level ones and uses the latter for encoding, which might improve the robustness of the AW representation to small temporal distortion.

The `other` layer is added to accommodate other functions related to AW extraction but do not belong to the other four layers. We consider the following three algorithms:

- **Normalization** is important for AW representations. For instance, it has been shown that taking the square root of $\boldsymbol{\alpha}$ (a.k.a. power normalization or probability product kernel transform) improves the linear discriminability of AW representations [12, 19].
- **Random sampling** exploits the repetitive nature of music signals and randomly samples (with replacement) the frame-level features of an audio clip to reduce the number of frames t to be encoded [19].
- **Consecutive frame concatenation** concatenates multiple vectors to capture temporal information [13]; can be performed after the `input` or `encoding` layer.

3. BENCHMARK

To validate the effectiveness of the AW representations in characterizing audio information, we compare eleven AW and non-AW features for three fairly different audio classification tasks. The features we consider include:

- **SPEC**: log-scale spectrum calculated with 1,024-point and half overlapping frames.
- **MFCC**: 20-D MFCC computed for each frame and aggregated by mean pooling. MFCC is included as the baselines.
- **AW-MFCC-VQ-REE** is extracted by using MFCC at the `input` layer, VQ and REE for encoding and dictionary learning at the `encoding` layer, and mean pooling at the `pooling` layer. The dictionary size is set to $k = 2,048$. A dictionary is constructed (in an unsupervised fashion) from the training and test sets of each of the three datasets, respectively, without assessing the class labels. The other seven AW representations, such as AW-MFCC-SC-REE and AW-SPEC-SCS-REE, are extracted similarly but using different algorithms at the `input` and `encoding` layers. However, an absolute-value `rectification` layer is applied to SC and SCS before pooling, because their encoding results have both positive and negative values. In addition, as SCS is more efficient than SC, we set $k = 16,384$ for SCS to evaluate the benefit of using a larger dictionary.
- **MIR** is a set of 177-D features representing the state-of-the-art "hand-crafted" music/audio features calculated by using the MIRtoolbox [10]. It includes two measures of energy, five descriptors of rhythm (e.g., average tempo, pulse clarity and event density), 148 timbre/spectral descriptors (e.g. MFCC, spectral centroid, flux, roughness, irregularity and zero-crossing rate) and 22 tonal descriptors (e.g. pitch class profile, key clarity and musical mode). It is considered as a strong baseline in that it captures not only timbre but also the rhythmic and harmonic aspects of music signals, whereas the AW representations we consider are based on only spectral features (i.e. MFCC or SPEC).

Prior to feature extraction, all audio clips are converted to 22,050 Hz single channel waveform.

3.1 Audio Classification Tasks

We perform the benchmark using the following three data sets to ensure the result is generalizable.

- **FreeSound** consists of 20,626 audio clips collected from Freesound (`http://www.freesound.org`). Each clip is manually labeled with one of the following five categories: `sound effect`, `soundscape`, `speech`, `instrument sample` and `complex music fragment` [6]. As the clips are annotated with mutually exclusive labels, we formulate the problem as a multi-class classification one and report the average classification accuracy (ACC). We consider the first 30s for overly long clips for feature extraction and use ten-fold cross validation (CV) for evaluation.
- **CAL** is a subset of the dataset collected by Tingle *et al.* [16] for music auto-tagging, which contains the genre and acoustic annotation of music labeled by professional music editors of the music service company Pandora (`http://www.pandora.com`). With the 7digital API (`http://www.7digital.com`) we collect the 30s audio previews of 7,799 songs, spanning 140 genres and sub-genres, such as `classical`, `rock`, `teen pop`, `motown`, `salsa`, `latin` and `deathcore metal`. We consider it as a multi-label problem and evaluate the precision for tag-based retrieval using the training/test splits (akin to five-fold CV) defined by [16]. The performance measures are mean average precision (MAP), precision at rank ten (P@10) and precision at rank R (P@R), R being the number of relevant clips [16].

Table 1: The results of the benchmark, with the top two results for each performance metric highlighted.

Feature	Dimension	FreeSound	CAL			MER		
		ACC	MAP	P@10	P@R	MAP	P@10	P@R
MFCC	20	0.365	0.067	0.076	0.069	0.021	0.026	0.031
MIR [10]	177	**0.551**	**0.153**	**0.188**	**0.164**	0.043	0.079	0.065
AW-MFCC-VQ-REE	2,048	0.459	0.076	0.110	0.085	0.075	0.235	0.130
AW-MFCC-TC-REE	2,048	0.479	0.106	0.139	0.120	0.038	0.081	0.060
AW-MFCC-SC-REE	2,048	0.465	0.083	0.113	0.093	0.079	0.258	0.135
AW-MFCC-SCS-REE	16,384	0.450	0.074	0.105	0.082	0.105	**0.314**	0.154
AW-SPEC-VQ-REE	2,048	0.497	0.109	0.149	0.120	0.063	0.179	0.122
AW-SPEC-TC-REE	2,048	0.533	0.138	0.170	0.149	0.044	0.070	0.058
AW-SPEC-SC-REE	2,048	**0.545**	**0.143**	**0.188**	**0.157**	**0.111**	0.300	**0.171**
AW-SPEC-SCS-REE	16,384	0.530	0.120	0.164	0.135	**0.111**	**0.323**	**0.162**

- **MER** contains 31,427 30s audio previews labeled with 190 music emotion tags [18] by the crowd of last.fm users (`http://www.last.fm`). These are the songs considered as most relevant to the emotion tags according to the `Tag.getTopTracks()` function of the last.fm API. We consider the subset of 43 emotion tags which appear in the Affective Norm for English Words [1], such as `sad`, `lazy`, `relaxed`, `happy`, `romantic`, `fun` and `angry`, and evaluate the accuracy for tag-based retrieval using the training/test split specified in [18].

We use the simple l_2-regularized l_2-loss linear support vector machine (SVM) algorithm implemented by LIBLINEAR [5] for classifier training and prediction. The SVM parameter C is tuned by an inner CV on the training set. We use power normalization for AW representations and z-score normalization for the MFCC and MIR features.

3.2 Results

Table 1 shows the results of this benchmark, from which the following observations can be made. First, generally speaking MIR features and SC-based AW features achieve the best results for all the three tasks, outperforming MFCC by a large margin. Moreover, while MIR features perform slightly better for the first two tasks, SC-based AW features lead to significantly better precision than the rivals for the MER task. This validates the effectiveness of an AW representation in representing audio information, even though it is based on only primitive spectral features randomly drawn from an audio collection and is computed without exploiting specific knowledge of the signal (i.e. music theories). The P@10 for MER reaches 0.323, which shows out of the top 10 ranked songs for each tag on average more than 3 of them are relevant. Second, by comparing the AW features, we see that generally better results for the tasks we considered are obtained by using SPEC rather than MFCC for the `input` layer, and SC-based algorithms (i.e. SC and SCS) rather than VQ or TC for the `encoding` layer. This shows the importance for the AWtoolbox to accommodate a variety of AW extraction pipelines to be experimented with by a user. Finally, according to our evaluation it is sufficient to use a dictionary of medium size (i.e. 2,048 for AW-SPEC-SC-REE) instead of a large one. It takes around 17 seconds for computing the AW-SPEC-SC-REE for a 30 seconds music clip.

4. CONCLUSIONS

In this paper, we have presented the AWtoolbox to facilitate the use of various AW features in representing audio information. The toolbox is characteristic of its flexibility in designing new AW extraction pipelines and its expandability in including new algorithms. The effectiveness of the resulting feature representations have been validated through three fairly different audio and music classification tasks. We hope that the AWtoolbox can also be useful for other tasks such as audiovisual video content analysis, audio-based duplicate detection (e.g. cover song identification), similarity estimation, visualization, and reconstruction-related tasks such as source separation and audio synthesis.

5. REFERENCES

[1] [Online] `http://csea.phhp.ufl.edu/media/`.

[2] F. Bach et al. Optimization with sparsity-inducing penalties. *FTML*, 2012.

[3] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, pages 921–928, 2011.

[4] L. Deng and X. Li. Machine learning paradigms for speech recognition: An overview. *TASLP*, 21(5):1060–1089, 2013.

[5] R.-E. Fan et al. LIBLINEAR: A library for large linear classification. *JMLR*, 2008.

[6] F. Font et al. Audio clip classification using social tags and the effect of tag expansion. In *Semantic Audio*, 2014.

[7] P.-S. Huang et al. Pooling robust shift-invariant sparse representations of acoustic signals. In *Interspeech*, 2012.

[8] P.-K. Jao et al. Modified LASSO screening for audio word-based music classification using large-scale dictionary. In *ICASSP*, 2014.

[9] Y.-G. Jiang. SUPER: Towards real-time event recognition in internet video. In *ICMR*, 2012.

[10] O. Lartillot and P. Toiviainen. A Matlab toolbox for musical feature extraction from audio. In *DAFx*, 2007.

[11] J. Mairal et al. Online dictionary learning for sparse coding. In *ICML*, pages 689–696, 2009.

[12] B. McFee et al. Learning content similarity for music recommendation. *TASLP*, 20(8):2207–2218, 2012.

[13] J. Nam et al. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, 2012.

[14] E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439(7079):978–982, 2006.

[15] L. Su, C.-C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang. A systematic evaluation of the bag-of-frames representation for music information retrieval. *TMM*, 2014.

[16] D. Tingle et al. Exploring automatic music annotation with "acoustically-objective" tags. In *MIR*, 2010.

[17] Z. J. Xiang et al. Learning sparse representations of high dimensional data on large scale dictionaries. In *NIPS*, 2011.

[18] Y.-H. Yang and J.-Y. Liu. Quantitative study of music listening behavior in a social and affective context. *TMM*, 15(6):1304–1315, Oct 2013.

[19] C.-C. M. Yeh et al. Improving music auto-tagging by intra-song instance bagging. In *ICASSP*, 2014.