ECHOPRINT - AN OPEN MUSIC IDENTIFICATION SERVICE

Daniel P.W. Ellis
LabROSA
Columbia University
dpwe@ee.columbia.edu

Brian Whitman
The Echo Nest
brian@echonest.com

Alastair Porter
CIRMMT / Schulich School of Music
McGill University
alastair.porter@mail.mcgill.ca

ABSTRACT

We discuss Echoprint, an open source and open data music identification service available to anyone. Echoprint is efficient and speedy and works by generating dozens of hashes a second from input audio (microphone or files) and then matching those hashes in a large scale inverted index for queries. We discuss the signal code generator and the server component. [1]

1. MUSIC FINGERPRINTING

"Fingerprinting" of audio files [1, 2, 4] is becoming a necessary feature for any large scale music understanding service or system. Online music stores want to resolve existing user catalog against their cloud storage to save bandwidth. Music indexing tools want to adjust poor metadata.

Last year we presented the Echo Nest Musical Fingerprint (ENMFP) [1], which was based on the detailed analysis of musical audio performed by the Echo Nest Analyze service [3]: fingerprints were based on the chroma vectors of several successive segments. While effective for matching different encodings of the same track, ENMFP could not handle more drastic spectral distortions such as are encountered in over-the-air (OTA) recordings. Also, since it relied on the output of the Analyze processing, it was computationally expensive if Analyze had not already been performed.

Echoprint is designed to resolve these shortcomings. It dispenses with the chroma features, relying only on the timing of successive beat-like events, a feature that remains robust under a wide range of channels and noise conditions. It also uses its own onset detection scheme which is far simpler than the one in Analyze, while remaining sufficient for the fingerprinting task.

[1] Ellis and Porter are paid consultants to The Echo Nest.

In the next section we present more detail on how fingerprint codes are generated from the source audio. In the following section, we describe the indexing and matching scheme in the server. Finally, we present some results.

2. GENERATING CODES FROM AUDIO

To achieve greater robustness to the kind of spectral modifications and noise encountered in OTA recordings, Echoprint relies only on the relative timing between success beat-like onsets detected in the audio. Onset detection is performed independently in 8 frequency bands, corresponding to the lowest 8 bands in the MPEG-Audio 32 band filterbank (hence, nominally spanning 0 to 5512.5 Hz). The magnitude of the complex band-pass signal in each band is compared to an exponentially-decaying threshold, and an onset recorded when the signal exceeds the threshold, at which point the threshold is increased to 1.05 $\times$ the new signal peak. An adaptive algorithm takes a target "inter-onset-interval" (IOI), and decreases the threshold decay rate when actual IOIs are shorter, or increases the rate of decay if they are longer. The target onset rate for Echoprint is 1 onset per second per band.

Pairs of successive IOIs in each band, quantized into units of 23.2 ms, are combined to make a hash. To provide robustness against spurious or missed onsets, each onset is considered along with its four successors. Six different hashes (IOI pairs) are created by choosing all possible pairs of succeeding onsets from the four.

Thus, the overall hash rate is approximately 8 (bands) $\times$ 1 (onset per second) $\times$ 6 (hashes per onset) $\approx$ 48 hashes/sec. Since onsets are approximately one second apart, the quantized IOIs are of the order of 1/.0232 = 43, or around 5-6 bits, and a pair of onsets constitutes around 12 bits of information. This is combined with the 3 bit band index to generate the raw hash, which is then stored along with the time it occurs within the file.

To improve robustness against varyiable OTA channel characteristics, the signal is "whitened" prior to the analysis above. A 40-pole LPC filter is estimated from the autocorrelation of 1 sec blocks of the signal, smoothed with an 8 sec decay constant. The inverse (FIR) filter is applied to the

signal to achieve whitening. Thus, any strong, stationary resonances in the signal arising from speaker, microphone, or room in an OTA scenario will be moderated by a matching zero.

3. MATCHING QUERY CODES TO SONGS

We have a database of known tracks, with each track T consisting of an ID and metadata (artist, album, track name). Performing a match involves taking an unknown audio query Q and finding the corresponding track in the reference database.

To build the database, each track is split into 60 second segments, with adjacent sections overlapping by 30 seconds. This helps to remove bias introduced when longer songs provide more matches for a set of query hashes.

The codes for a 60 second segment are represented as terms of a document D in an inverted index. The combination of the unique track ID plus the segment number is used as the document ID. Our underlying data store uses Apache Solr with a custom query handler to provide a fast lookup of a code query to list of document IDs.

A 30 second query has about 800 hash keys. The query server returns the documents with the most matches of each code term in the query. In practice we find that there is rarely one document with significantly more matches than all other documents in the index, however, the top matches (we use 15) in this metric will contain the actual match if it exists. We compute a histogram of all time offset $\mathbf{t}$ differences per matching key in the result set. We then use the total of the top two histogram buckets to inform the "true score." This allows us to ensure that the codes occur in order even if Q is from a different section of the song and thus has a different absolute time offset.

All possible documents are ordered by their true score. If more than one document from the same track are in the list, we remove all but the document with the highest score. The top document in the list is returned as a positive match if its true score is significantly higher than the score of all other documents in the result list. If the gap between the top two results is insignificant then no match is returned.

4. DATABASE SCALING AND EVALUATION

ENMFP [1] quickly grew to over 30 million tracks within a matter of months and Echoprint will also see a strong growth pattern given that the data will come from outside sources. We engineered the database layer of Echoprint to handle hundreds of millions of tracks by splitting apart the index (the inverted Solr-backed document store) and the storage layer. The index takes roughly 5 gigabytes of disk space per 100,000 tracks, while the storage requires 15 gigabytes. Queries can be executed in practice in 100 milliseconds after computing the code string for a signal, which on current hardware takes under a tenth of a second.

To evaluate Echoprint we built a suite of tools that would test various permutations of queries. We use a test set of 100,000 tracks alongside a set of 2,000 tracks that are known to be not in the set of 100,000. We compute queries on audio pulled from selections of the audio files, for example, 30 seconds from the middle, downsampled to 96kHz or decoded using various MP3 decoders or had their volume adjusted. This lets us compute metrics such as the number of false positives fp (the wrong song was identified), false negatives fn (the query was not matched to the correct track), false accept fa (a known non-database query was matched to a track,) true positive tp and true negative tn. Using these measures we compute a probability of error $P(E)$ that weights the size of the database (100,000 as D) and the size of the known-non-database tracks (2,000 as N) with the false reject rate R_r and the false accept rate R_a:

$$P(E) = ((\frac{D}{D+N}) * R_r) + ((\frac{N}{D+N}) * R_a) \quad (1)$$

where

$$R_r = \frac{fp + fn}{tp + fp + fn} \quad (2)$$

$$R_a = \frac{fa}{fa + tn} \quad (3)$$

We have published results [2] of various $P(E)$:

Manipulation	$P(E)$
30 second WAV file	0.0109
60 second WAV file	0.0030
60 second recompressed MP3 file	0.0136
30 second recompressed MP3 file	0.0163
30 second recompressed MP3 file at 96kbps	0.0260

5. REFERENCES

[1] Daniel P.W. Ellis, Brian Whitman, Tristan Jehan, and Paul Lamere. The Echo Nest musical fingerprint. In *Proceedings of the 2010 International Symposium on Music Information Retrieval*, 2010.

[2] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system with an efficient search strategy. *Journal of New Music Research*, 32(2):211–221, 2003.

[3] Tristan Jehan. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, 2005.

[4] A. Wang. An industrial strength audio search algorithm. In *International Conference on Music Information Retrieval (ISMIR)*.

[2] http://echoprint.me