Online music recognition: the Echoprint system

Cors Brinkman
Leiden University
cors.brinkman@gmail.com

Manolis Fragkiadakis
Leiden University
vargmanolis@gmail.com

Xander Bos
Leiden University
xander.bos@gmail.com

ABSTRACT
Echoprint is an open source music identification system build by The Echo Nest. This paper discusses the workings of the Echoprint system. The paper explains topics such as the purpose, context, history and operating principles of Echoprint, combined with the strengths, weaknesses and both intended and surprising applications of the system. These topics are discussed so that a proper overview of the system is drawn. Interested readers can follow the process in the getting started section, which explains briefly how to set up the system and how to use it. The paper is concluded by some final thoughts about the Echoprint system and its future.

Figure 1. The Echo Nest logo

1. PURPOSE, CONTEXT AND HISTORY

Echoprint [4] is a music identification system build by The Echo Nest [9], which utilizes acoustic fingerprinting technology for the identification functionality. Think of actual human fingerprints being used for identification. Acoustic fingerprinting uses the same principle, but by means of audio. Echoprint consists of a set of components which can be used to either experiment with and/or set-up an audio identification system/service. An acoustic fingerprint is achieved by creating a condensed digital summary of the audio signal. The system "listens" to the audio signal being played. An algorithm places marks on, for example, different spikes in frequency of the signal and is able to identify the signal by matching these marks to a database on an external server. The signal can include all different forms of audio, songs, melodies, tunes and for example sound effects from advertisements. A more in detail description will be given later.

By developing an identification system like Echoprint, it is made possible to monitor the 'usage' of music online. Similar systems have been used in many different cases of music classification, copyright detection, measuring usage of music on peer to peer networks and for creating catalogues in the music industry. The main purpose of Echoprint is, however, music recognition. The audio data for recognition can be provided by all kinds of platforms, but comes mainly from phones or computer systems. After processing an audio signal, Echoprint can tell what song and kind of music is currently playing and reports the match back to its user. The Echoprint system is open-source and installable on private servers. The database needed for the identification of music is also made openly available and downloadable by the publisher itself, although it's currently outdated.

The Echo Nest is a music intelligence and data platform for developers and media companies. Based in Somerville, MA. The Echo Nest was originally a research spin-off from the MIT Media Lab to understand the audio and textual content of recorded music. Its creators intended it to perform music identification, recommendation, playlist creation, audio fingerprinting, and analysis for consumers and developers. Leading music services (Clear Channel's iHeartradio, MOG, Rdio, SiriusXM, Spotify), editorial, video and social media networks (BBC.com, Foursquare, MTV, Twitter, VEVO, Yahoo!), connected device manufacturers (doubleTwist, Nokia) and big brands (Coca Cola, Intel, Microsoft, Reebok) together, resulting in over 100 million worldwide users of The Echo Nest platform. [6]

- The Echo Nest was founded in 2005 from the dissertation work of Tristan Jehan and Brian Whitman at the MIT Media Lab.
- In June 2011, the company released Echoprint, an open source and open data acoustic fingerprinting system.
- On March 6, 2014 Spotify announced that they had acquired The Echo Nest.

2. OPERATING PRINCIPLES

Echoprint's ability to identify a music track substantially fast and with high accuracy, makes it one of the most valuable music identification systems available. Additionally, it can even recognize noisy versions of the original track and even recordings performed by mobile devices with noise "bleed" by environmental factors.

The three main parts of Echoprint's architecture are (figure 2):

- The code generator: responsible for the audio-to-code

conversion

- The server: capable for storing and indexing the code
- The data: gathered from other Echoprint users and partners

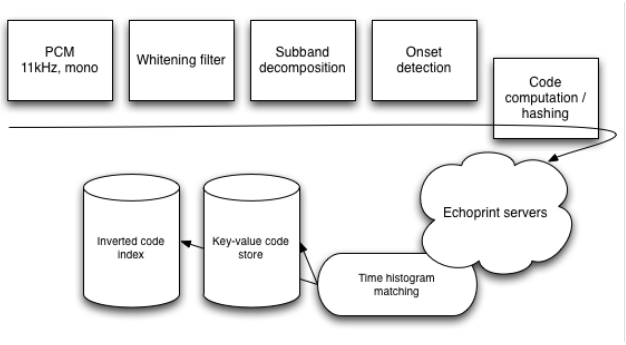

Figure 2. Schematic of the technology

The code generator, using advanced signal processing, "fingerprints" audio data. Echoprint captures the relative timing between success beat-like onsets detected in the audio, creating {time, hash} pairs. It captures the peaks in the spectrogram and stores them based on their time of occurrence (figure 3). Each song then can be identified by these pairs since they are unique.

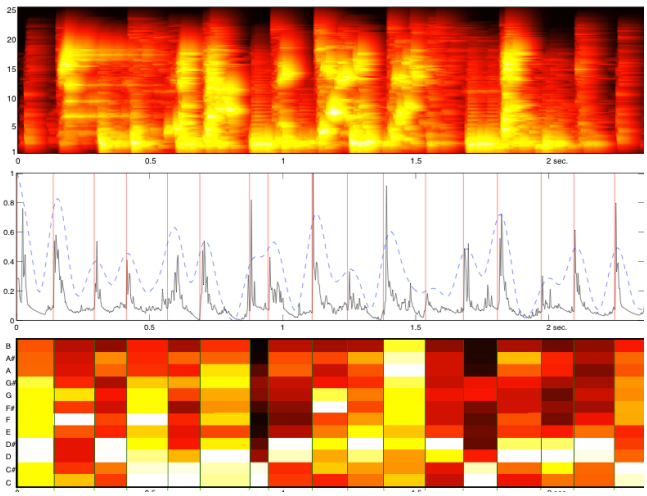

Figure 3. Onset detection on the spectrogram

First it computes a whitening filter based on an 11Kz mono signal which is capable of reducing the noise from the environment and lead to a "clean" recording. Then, the onset detection is performed on the lowest 8 bands in the MPEG-Audio 32 band filterbank (subband decomposition). This is done because it is easier for the algorithm to detect the onsets on lower frequencies. The magnitude of the signal in each band is compared to a predefined exponentially-decaying threshold. When that threshold is exceeded, an onset is recorded. After that the threshold is adapted to match the new signal peak. Subsequently, the onset material is hashed into a 20 bit space and stored alongside the time of the onset. The code generator then pushes that information into the database for comparison.

The server stores the occurrence of the onset in a list and indexes each onset in an inverted index. The database contains predetermined tracks with their ID and their metadata (artist, album, track name). Each track is splitted into 60 seconds segments and the code for each segment is represented as terms of a document ID in an inverted index. The combination of the unique track IDs plus the segment number is used as the document ID. Then, the querying is performed by looking up all the queries in the inverted index and returning a score of overlapping onset queries between the query and each target track. The score determines whether there is a match and how similar the songs are.

3. STRENGTHS AND WEAKNESSES

According to J. Haitsima and T. Kalker the definition of an audio fingerprinting system relies on its robustness, reliability, fingerprint size, granularity, search speed and scalability [12].

It can easily be deduced that the powerfulness of Echoprint lies on its fast and accurate recognition. The overall architecture provides an over-the-air song detection with the advanced signal processing accounting the noise from outside sources and the medium of transportation. Moreover, it can identify remixes, live versions and sometimes even covers from the original song while the time needed to do such an identification is even less than 20 seconds of recording. The open source server and part of the code generator allow developers to further explore the potential possibilities.

However, Echoprint isn't a fully open source system, because the code generator relies on proprietary Echonest's algorithms that are essential part of the identification and classification procedure. The difficulty on the configuration of the API and the database adds to the otherwise few weaknesses of the system.

Some other technologies include Shazam, Soundhound and MusicBrainZ. Shazam is the most well know music identification application that is significantly similar to Echoprint. With its well built user interface and its prevalent presence in the smartphone world, it is considered as Echoprint's biggest competitor. While it is easy to use and has already an application, its API is not public and does not allow modifications. Similarly, Soundhound and MusicBrainZ provide a mobile music and audio recognition service with the first one being able to even identify music through singing or humming and the later providing an open content music database.

The final word is up to the users of whether they prefer to use an overall end product or discover and further explore Echoprint and modify it according to their preferences and demands.

4. TYPICAL APPLICATIONS

Typical applications for Echoprint are straightforward, music recognition and digital music advertising in other systems such as Spotify, but also in commercials or products of a wide range of worldwide corporations. Echoprint correctly adds the metadata and indexes playlists of different online radio, video and streams.

Other well known and typical applications of acoustic fingerprinting systems are applications of competitors like Shazam and Soundhound, as discussed in section *3. Strengths and weaknesses*. Open the app on your phone and record a few seconds of the music you are currently listening to or simply hum it. Their databases returns, in the ideal scenario, the information of the song you want and are looking for.

5. SURPRISING APPLICATIONS

During the research on Echoprint, some surprising applications were encountered. A good example is the fact that the system is being used for copyright detection. Even more surprising is the fact that huge platforms such as Spotify and Pandora use the Echoprint system to index their online playlists and radio-streams. By indexing the songs, playlists and queries, a valid estimation of usage of songs can be measured.

Another surprising and creative application of the Echoprint system is the 3D music maze (see figure 3)[1]. This app is an experiment from The Echo Nest lab in using alternative interfaces for music exploration and discovery. The user can walk through a generated maze of 3D album art boxes. By walking close to the album art, the music of the album is being played. It uses The Echo Nest artist similarity and playlisting APIs to build logical clusters of artists and songs. It uses the 7Digital media [2] for the album art and 30 second samples and three.js [8] for all the 3D modelling.

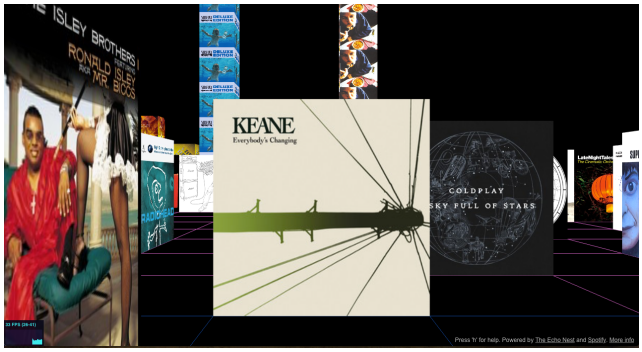

Figure 4. The Echo Nest lab, Maze experiment

The Echoprint system has also been used for further research in the field of audio recognition. For instance on predicting music taste by movie taste and vice versa[7].

6. GETTING STARTED

In this section it will be explained how to get started with the Echoprint system. There are two important elements that are required to fully utilize this system. For the both of them, it will be briefly explained how to set them up and in more detail it will be explained how to use them.

6.1 FIRST PART - CODEGEN PROGRAM [3]

The first element is the codegen program. This program is written in C++ and should compile/work on all major platforms. Through a command line interface, it takes an audio fragment to generate a special code which later can be used to either store or find matching sounds. There are no binaries available for download, so it has to compiled by hand. To do this, a set of dependencies/tools are required to be available. These are listed here:

- CMake; a software that enables build automation, similar to Linux's make program
- FFmpeg; a software that is widely used for audio- and video conversion
- Boost++; a library that makes C++ a more productive programming language
- Taglib; a library that enables reading and editing of metadata in popular multimedia formats
- Zlib; a library used for data compression

In the documentation it rather clearly explained how all of these are used. The codegen program can generate codes in two different ways. It can either generate a code based on a single file or work through a list of files – outputting the result in JSON format.

The command is structured as follows:
codegen.exe [filename | -s] [seconds_start] [seconds_duration] [< file_list (if -s is set)]

Additional parameters are [seconds_start] and [seconds_duration]. These parameters are used to determine what part of a sound file will be used to generate a code. [seconds_start] is used to set a starting point for analysing and [seconds_duration] determines how many seconds after the starting points are being taken into account. Also note that the brackets are not used in actual commands.

A simple usage example:
codegen.exe audio_file1.mp3 0 30

In this command the first thirty seconds of the audio_file1.mp3 will be analysed. After analysing it will output the code, which is a long single line of text consisting of letters and numbers. Its length is dependent on how many seconds it had to analyse. How this code can be used will be explained in the next part.

6.2 SECOND PART – SERVER PROGRAM [5]

The second element is the server. The server consists of a few parts, each written in a different programming language. These are the parts:

- Echoprint custom component for Apache Solr
- Tokyo Tyrant database
- Python-based API layer

The Apache Solr part is used to keep indexes of the Echoprint codes. The actual data that belongs to an index is stored in the Tokyo Tyrant database. The API layer contains the logic for finding and inserting audio. It should be noted that the Tokyo Tyrant database is supported on Linux systems only, which effectively makes the server only runnable in a Linux environment. And like the codegen software, the server has some dependencies:

- Java; a well-known runtime environment which in this case is required for Apache Solr
- Tokyo Cabinet; a library required for Tokyo Tyrant to run
- Python; an interpreted language which is used for the API layer
    - Web.py; an extension that allows for a HTTP request based API

Installing these dependencies is relatively easy as they're well-documented. To get the server running, the Apache Solr server and Tokyo Tyrant database server must be started. To enable access to them, the api.py program must be run – which is included in the Echoprint server "package". Now we access to two commands. Ingest, which is used to add new audio tracks to the database and query, which is used to find matching sounds. The ingest command is a HTTP POST request.

The URL that will trigger the commands will look similar to this example:
http://localhost:8080/ingest

The POST body has the following parameters:

- fp_code; the code generated by the codegen program
- track_id; optional parameter that allows you to give your track a specific ID
- length; the length of the audio fragment analysed in seconds
- codever; the version of the codegen software used to generate to code
- artist; optional value to specify the name of the artist of the audio fragment
- release; optional value to specify the release of the audio fragment (e.g. album)
- track; optional value to specify the name of the audio fragment

After a successful request, a response will be returned in JSON format. The JSON will hold a status code and a track ID. The track ID should match the parameter you send, unless you omitted it. The query command can be either a HTTP POST or GET request. It only has one parameter, which is fp_code. This code is, like the parameter from the ingest command, generated by the codegen program.

When using GET, the URL would look similar to the following example:

http://192.168.145.128:8080/query?fp_code=eJydz1uKgCEIBeAtaZbZcrzk

Upon a successful request, a JSON will be returned. This JSON will contain more information in comparison to the ingest command. The most interesting information is whether the query was executed correctly, the search time, if a match was found and if it did find a match, it will also contain the track's ID. The API can be easily extended so that it will also return the track's metadata.

7. FINAL THOUGHTS

Overall the Echoprint system seems reliable and accurate compared to it's competitors. It can be considered as a milestone in the development of audio fingerprinting technologies and the expectations of its developers to be the de facto music identification technology, seems valid. Its speed and accuracy provides a technology that can be further developed and explored, as it is already been done by major corporations in the field in discussion. The Echoprint's rich data API and developer toolkits power the most engaging music applications in the industry.

Echoprint, from a development perspective, is very interesting to work with, but also very time-consuming. This is especially true when you don't have some experience with C-based programming languages, as some components need to be compiled. It took five working days to get everything set-up and running. Most of the encountered problems were caused by memory leaks or bad documentation. On the Echoprint forum these problems are discussed and usually a possible solution is provided.

The interesting part is that Echoprint is open-source. This means that the system is extendable. Mainly the API, as it is easy to understand. To add more functionality it would require a deep understanding of the workings of the system, which is achievable by analysing the code.

REFERENCES

1. "3D Music Maze." 3D Music Maze. Web. 12 June 2015. <http://labs.echonest.com/3dServer/maze.html>."
2. "7digital United Kingdom | High-quality Digital Music Downloads." 7digital. Web. 2 June 2015. <https://www.7digital.com>.
3. "Echoprint Codegen Instructions." GitHub. Echo Nest,

9 Jan 2015. Web. 12 June 2015. <https://github.com/echonest/echoprint-codegen>.

4. "Echoprint." Open Source Music Identification. Web. 20 Apr. 2015. <http://echoprint.me>.
5. "Echoprint Server Instructions." GitHub. Echo Nest, 10 May 2012. Web. 12 June 2015. <https://github.com/echonest/echoprint-server>.
6. "Our Company." Company. Web. 12 June 2015. <http://the.echonest.com/company/>.
7. "The Echo Nest Releases New Research Findings." The Echo Nest Releases New Research Findings. Web. 12 June 2015. <http://the.echonest.com/pressreleases/echo-nest-releases-new-research-findings/>.
8. "Three.jsr." Three.js. Web. 2 June 2015. <http://threejs.org>.
9. "We Know Music..." The Echo Nest. Web. 12 June 2015. <http://the.echonest.com/>.
10. Bertin-Mahieux, Thierry, et al. "The million song dataset." ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida. University of Miami, 2011.
11. Ellis, Daniel PW, Brian Whitman, and Alastair Porter. "Echoprint: An open music identification service." ISMIR 2011 Miami: 12th International Society for Music Information Retrieval Conference, October 24-28. International Society for Music Information Retrieval, 2011.
12. Haitsma, Jaap, and Ton Kalker. "A Highly Robust Audio Fingerprinting System." ISMIR. Vol. 2002. 2002.
13. Schindler, Alexander, and Andreas Rauber. "Capturing the temporal domain in echonest features for improved classification effectiveness." Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation. Springer International Publishing, 2014. 214-227.