

Music Inspector: A Framework for Audio Copyright Infringement Tracking

Jasmeet Jagdev

Madhur Kukreti

Disney Y. Lam

1.0 Introduction

The advent of the Internet has brought convenience in how we share content; however, it has also brought tremendous challenges for those who used to profit from the content they author or own. For the music industry, copyright owners struggle with many individuals downloading their content without permission, nor compensation. This problem is difficult to solve, since it is hard to target many individuals who are geographically dispersed, not to mention that most individuals are commoners who do not possess many assets to sue for in the first place. Moreover, in Canada, copyright owners are responsible for finding and reporting the copyright infractions [1]. In this paper, we define copyright infringement of music similar to that of blatant academic plagiarism; that is to say, using any part of a copyright owner's original material without compensation and/or acknowledgement. Due to differing political and demographic beliefs regarding how pirating music should be dealt with, we leave this detail to the discretion of the copyright owners and legal system. However, we do believe that copyright owners should have the right to know where their content is being hosted for illegal download, so that they can take actions suitable for their environment. In the subsequent sections, we present Music Inspector, a framework that enables copyright owners to discover Internet sources that are distributing their content without permission. We then demonstrate the effectiveness of our framework in tracking audio copyright infringement through an extensive example. Finally, we discuss limitations, provide recommendations, mention future work, and conclude.

2.0 Solution

As previously introduced, Music Inspector assists copyright owners track where their content is being distributed without consent. The general mechanics are depicted in **Figure 1**. The frontend consists solely of the user interface module, used to facilitate user input and output to and from the backend. The backend consists of two modules: crawler and acoustic verification. The crawler module searches the web for potential sources of plagiarism of the original audio file supplied by the user interface input and removes whitelisted sources. The audio verification module compares original content with that found from the crawler module. Finally, the list of sources with copyright infringements is outputted back to the user interface module, with the option of contacting copyright infringed content hosts via email. Details for each of these components is further discussed in the following subsections.

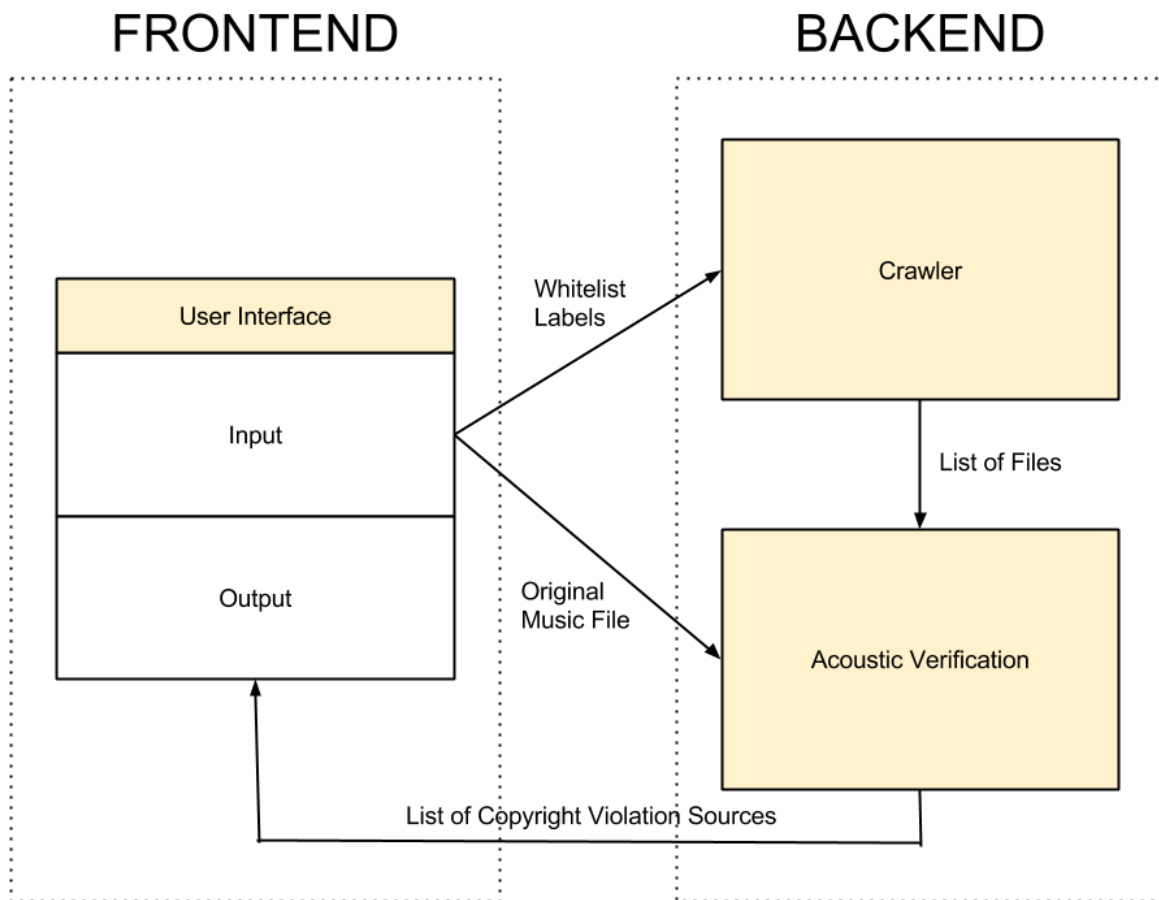


Figure 1: System Flow Diagram of Music Inspector

2.1 Crawler

The main purpose of the crawler module is to search for potential sources where copyright infringements have occurred. This is achieved in a series of steps with the help of several components, as shown in **Figure 2**. Please note that the boxes with dotted lines are inputs from the user interface module and the bolded box is the output given to the acoustic verification module.

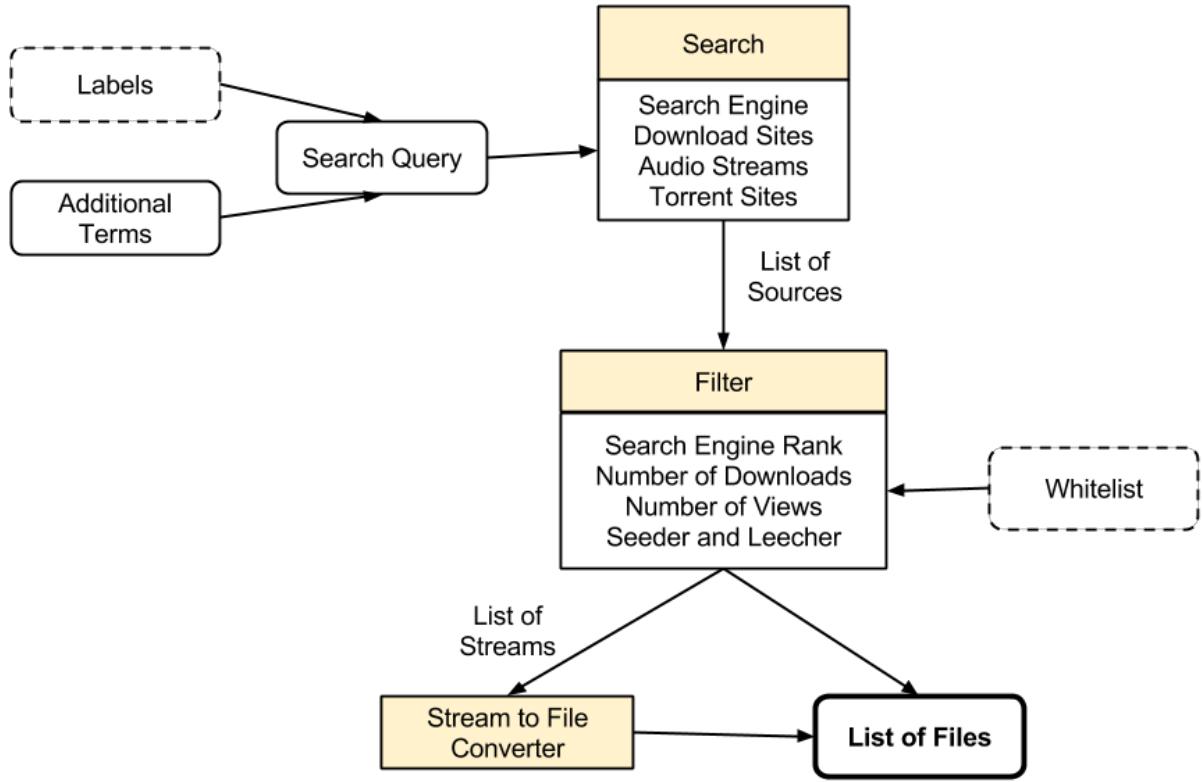


Figure 2: Component View of the Crawler Module

As previously mentioned, a copyright owner using our system inputs a series of labels associating his or her audio content with context and a whitelist. Labels include strings such as artist name, song title, and album title. The labels are then used to search the web. In order to specifically target copyright infringement sources, our system appends additional terms to the labels. The terms that we are referring to include words that people would add to a search query to pirate a song or album, such as “torrent” and “free download”. Suppose there are l labels and t additional terms. We denote each label as L_i where i is in $[1, l]$ and each additional term as T_j where j is in $[1, t]$. Our system would produce the search query:

$$(L_1 \text{ or } L_2 \dots \text{ or } L_l) \text{ and } (T_1 \text{ or } T_2 \dots \text{ or } T_t).$$

This formulated search query is then supplied to the search module. Within the search module, we attempt to discover sources of copyright infringement using web services that either advertise the location of and/or host files for distribution. These include: search engines, torrent sites, download sites, and websites hosting streams. Since most of these services contain an indexed search engine to help users find content, we can leverage this to find potential stolen content using our search query. Due to the four different services used, we employ four ways to calculate the metric to determine the likelihood of high illegal audio file distribution.

Thus, the metrics are based on: search engine rank, number of seeders and leechers, number of downloads, and number of views. Particularly, search engine rank is a rank of how early a search result appears. We attempt to compute a metric that is directly associated with the number of people illegally hosting or obtaining the content. For download sites and audio streams, the metrics is calculated directly as the number of downloads and number of views respectively. For torrent sites, we calculate the metric as the sum of leechers and seeders.

For audio that have none of the statistics afore mentioned, we associate the metric of the content with that of the webpage it is found on. We then compute the metric to be the product of the inverse of the search engine rank and the search engine weight. Search engine weight is associated with the search engine queried and the number of users it serves. To determine specific values, experimentation will need to be done so that the values correspond to the number of users that will use the search engine result to access copyrighted material. For example, Google would receive a higher value than Ask.com, since more users use the former.

Since we want the metric to be directly correlated to the number of people accessing a file, we use search engine rank as a last resort when calculating the metric value. This is shown in **Algorithm 1**, where the `getMetric` function will return if torrent site, download site, or web stream information is available prior to using the search engine rank. If the metric calculated for a source is greater than a pre-determined threshold, it is considered a potential copyright infringement.

```
getMetric(Source s)
{
    if(seeder(s) and leecher(s) exists), return (seeder(s)) + (leecher(s));
    elif(numDownloads(s) exist), return numDownloads(s);
    elif(numViews(s)) exist, return numViews(s);
    else, return SearchEngineWt*(1/searchEngineRank(s));
}
```

Algorithm 1: Scoring and Weight for Potential Copyright Infraction Sources

In order to make the search and filter components scalable for the Internet, we limit the amount of information queried and the number of iterations when searching. In the search module, we accomplish this by allowing a maximum of f infringements to be found and limit the algorithm to search for infringement sources for at most n iterations, allowing for only p search results to be obtained per iteration. The combined function of the search and filter components is actually best described in an algorithm, which is presented in **Algorithm 2**.

```

// constants: f,n,p
// f: maximum number of infringements
// n: maximum number of search iterations
// p: maximum number of search results at each iteration

search&filter(string[] Labels, string[] AdditionalTerms, File Whitelist)
{
    int searchIteration = 0
    int numInfringements = 0;

    while(numInfringements < f and searchIteration < n)
    {
        infringementsFound = find searchIteration*p-th to (searchIteration+1)*p-th
        infringement sources using Labels and Additional Terms
        numInfringements += (infringementsFound – whiteListMatches)
        searchIteration ++;
    }

    return sources part of numInfringements count
}

```

Algorithm 2: Scalable Search and Filter for Copyright Infringement Sources

When the function in **Algorithm 2** produces a list of potential copyright infringement locations, we then proceed to download the files and streams. Audio streams are converted into audio files via conventional stream converters widely available on the Internet. Finally, this list of music files and its sources are passed onto the acoustic verification module.

2.2 Acoustic Verification

Just like a human fingerprint can be used to uniquely define a person’s identity, in a similar way acoustic fingerprints are used to uniquely identify sound. There are several ways we can use to compare two audio samples. First, we can directly compare their digitized waveforms. This method is trivial and inefficient since a small change in amplitude or frequency will cause different waveforms. A better way is to convert the audio samples into a compact binary representation by using hash methods such as MD5 (message digest 5) [4]. The problem with this method is that the audio samples have to be exactly same in every aspect. It is possible for two audio samples to have exactly same digitized waveforms but different sampling and compression rates. Moreover, hashes are very sensitive to change and a change in any of these values will completely change the hash. An effective acoustic fingerprinting system should be able to assign fingerprints based on the content or information in the song. It should overlook

compression levels, distortion and noise in the transmission channel. Important aspects that the system should be able to deal with are:

- ❖ Cropping or shifting
 - Audio samples used to find a match may contain lag i.e. If two exact songs are taken but one starts at 00:00 (mm:ss) and the other starts at 00:07, they might result in different fingerprints if the system does not deal with shifting. Also, we need to determine the minimum length of the audio input which is required to compute the fingerprint. This feature is also known as the granularity.
- ❖ Pitching
 - While mixing music and creating mashups, the pitch and speed of the songs have to be altered for smooth transitions and chord agreements. The system should be robust to these changes as well.
- ❖ Equalization
 - Equalization means altering certain frequencies in the spectrum without affecting the neighbouring frequencies. The most basic type of equalization familiar to most people is the treble/bass control on home audio equipment. The treble control adjusts high frequencies, the bass control adjusts low frequencies. This feature is generally used to make the audio more intelligible to the users. For an acoustic fingerprinting system, equalization should not affect the fingerprint.
- ❖ D/A-A/D conversions
 - When we record sound from our cell phone's built in recorder and store it in the memory, we are converting an analog signal into a digital (binary) signal. On the other hand, when we use our earphones to listen to that recording again, we are converting digital to analog. Digital and analog samples of the same song should result in similar fingerprints.
- ❖ Encoding schemes
 - Different audio and speech encoding schemes provide different sound quality, size and portability. Similar fingerprints should be generated for an audio sample across different formats and encoding schemes.

In order to match two pieces of music which are essentially recorded in different environments i.e. the two audio samples with different bit rates, audio formats, noise, amplitude etc. we need a fingerprinting mechanism which can extract the “perceptual” characteristics out of the audio even if they differ in their binary representation. In simple terms this means that if two audio samples sound similar to human ears they should have the similar acoustic fingerprint and if not, they should have different fingerprints. Consequently, acoustic fingerprints don't have to be extremely sensitive to variation which have insignificant impact on the features of the fingerprint. In a typical music identification scheme, the database of the music identification service consists of the fingerprints of the songs. Fingerprints are

computed in several stages. First, the input signal is transformed into a common digital format. This format has specified bitrates, sampling rates and encoding scheme. Therefore the format of the input signal is not relevant in spectrogram generation. The converted file is then used to generate a spectrogram. A spectrogram is used to get a visual representation of the spectrum of frequencies in the input signal that varies with time or some other variable. Next step in the process is band filtering. In band filtering the focus is narrowed down to parts of the spectrogram with “high energy content” i.e. it discards all the frequencies which contain little information and keeps the parts with frequency peaks. Band filtering enables us to focus on specific areas in the frequency spectrum. For example, if we need to focus on the bass line of a song, we can filter out higher frequencies and zoom in on the lower frequencies to analyse their wavelet patterns. Band filtering plays a major role in feature extraction. Once the information rich portions of the song are identified it is fed to a LSH (Locality Sensitive Hashing) scheme. In this phase, the input is hashed in such a way that similar inputs are mapped to the same buckets with a high probability.

If $d(p,q) < R$

Then $h(p) = h(q)$ with probability at least $P1$

And if $d(p,q) > R$

Then $h(p) = h(q)$ with probability at most $P2$

Here d is distance between p and q where p and q are two objects having similar perceptual features and R is the threshold distance. “ $h(.)$ ” is a typical hash function with the aforementioned properties. So the characteristics of a LSH scheme is that unlike the hash functions used in cryptography where we want to minimise collision, here we want to maximize collision in order to get all the similar hashes in the same bucket. By getting such a grouping, we can overlook changes in audio quality, amplitude and noise. This is exactly what we want since the recorded sample which needs to be identified is not only recorded by phone’s microphone which has a lower bitrate but also noise present in the samples is significant. We can choose from several hash functions such as MinHash, Bit sampling, Random projection etc. which have a property of LSH high collision. These fingerprinting hash functions often work on the derivative of the input rather than on their absolute values which means that they capture the changes in the input signal. This is because spectrogram of two similar signals will have similar derivatives. For example, a hash function can set the bits of a vector by comparing the current frequency with the subsequent frequency. If (next frequency > current frequency) it sets the bit to 1. If (next frequency < current frequency) it sets the bit to 2. If (next frequency == current frequency) it sets the bit to zero. In this way the hash obtained is a function of the “change” in frequencies rather than the absolute values of the frequencies. The final step of Acoustic Fingerprinting is fingerprint matching. Once we have a fingerprint database, the fingerprint of the incoming audio is computed and matched against the fingerprints in the database. If a match is found, all the relevant data about the song is fetched.

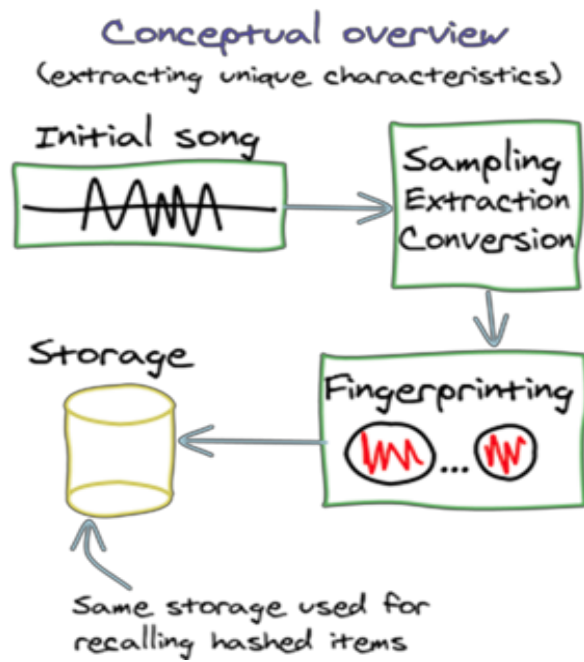


Figure 3 : Extracting Characteristics

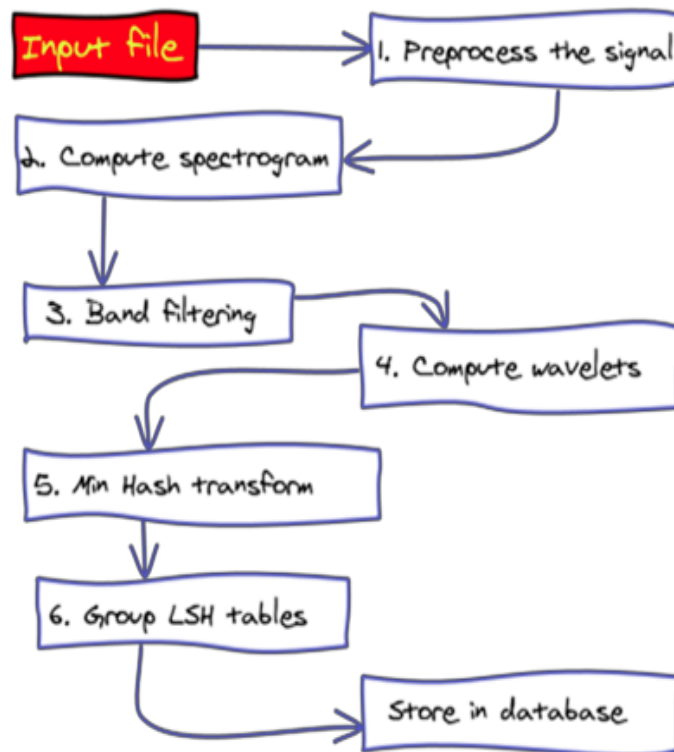


Figure 4 : Fingerprint generation

The two most popular music recognition mobile application on the google playstore - Shazam and Soundhound use similar mechanisms to identify audio. Shazam music identification service uses phone's built in microphone to record music samples which are of relatively low quality. This sample is uploaded to Shazam's web service and it then creates a "fingerprint" for the sample and matches it against the "fingerprints" of songs available in its own database that at present consists of 11 million songs. If a close match is found, Shazam pulls all the relevant information of that song i.e. song name, artist name, album name, year published and also generate links to the songs on various services such as Itunes, Zune, Spotify and Youtube. Applications like Soundhound on the other hand in addition to identifying songs using audio samples of the original track can also determine songs based on humming patterns of human beings. This is however much less accurate and also of little use to our system since we aim to find copies of the original audio.

Here, the acoustic fingerprinting system receives five files from the web crawler after whitelisting and filtering. The aim of this phase is to verify the files and confirm if they are actually a copy of the original song. The results of the acoustic verification phase is then presented back to the user of the system, as shown in **Figure 5**.

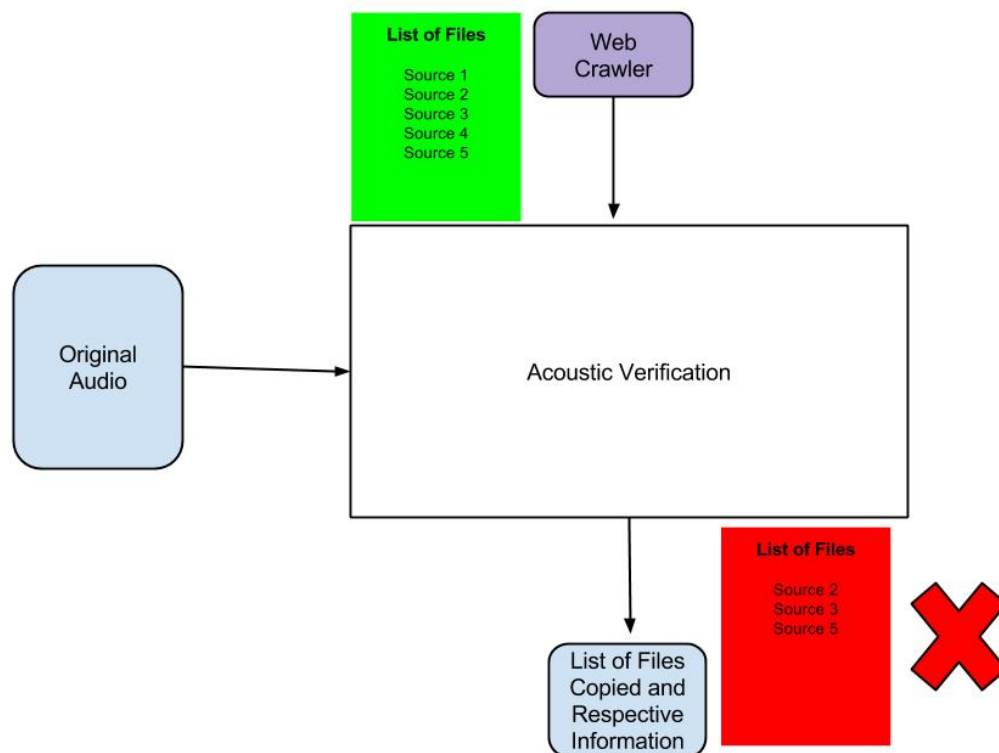
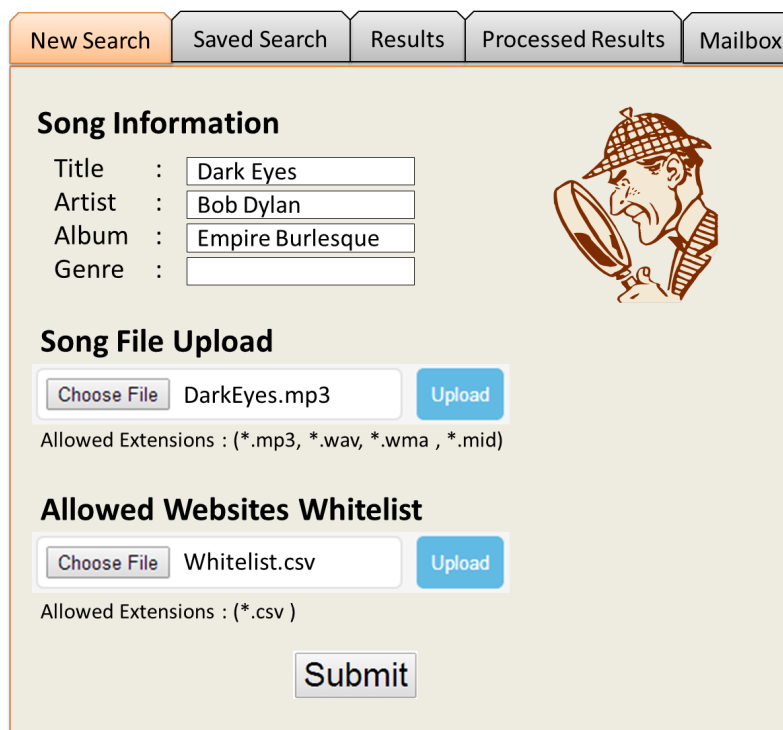


Figure 5 : List Generation

2.3 User Interface

Music Inspector acts as a platform for artists to identify websites that are under copyright violation. **Figure 6** shows the basic form in which user is provided to fill in the information. In general, it consists of three types of informa:

1. Basic information about the song, such as title, artist, album, and genre.
2. An option to upload the song.
3. An option to upload a whitelist of websites which have copyright license to the original content.



Song Information

Title : Dark Eyes
Artist : Bob Dylan
Album : Empire Burlesque
Genre :

Song File Upload

Choose File DarkEyes.mp3 Upload
Allowed Extensions : (*.mp3, *.wav, *.wma , *.mid)

Allowed Websites Whitelist

Choose File Whitelist.csv Upload
Allowed Extensions : (*.csv)

Submit

Figure 6 : New Search Tab for User Input

A. New Search tab

This is a place for the user to provide basic information about the song whose illegal copies needs to be detected. This information consists of commonly used tags for finding songs over the Internet, namely, song title, artist, album and genre. By doing this, we are trying to step in the shoes of a common user who tries to find free content on the web. This information is fed into the web crawler (see in **Section 2.1**), whose task is to use this information, match it over the internet, and form a list of potential links which appear to be copies of the provided song.

The second input is the original song, uploaded by the user. This will be used by the acoustic verification

system (see **Section 2.2**) to determine whether the target link is a copy of the original song or not. If it is found as a copy, then the information is recorded and presented in a tabular form in the results tab as shown in **Figure 7**.

The third piece of information provided by the user is the whitelist which has the names of the websites that are allowed to publish the copyright owner's content. That is to say, the names present in this list would be discarded from the results generated by the system.

The screenshot displays the 'Results' tab of a web application. At the top, there are five tabs: 'New Search', 'Saved Search', 'Results' (which is highlighted in orange), 'Processed Results', and 'Mailbox'. Below the tabs, there is a form area with a light beige background. At the top of this area is a label 'Owner Email ID : ' followed by a text input field. Below this is a section titled 'Copyright Violations'. Inside this section is a table with two columns: 'Website' and 'Link to Song'. The table contains three rows of data. Each row has a checkbox to its left. Below the table is a button labeled 'Send Notification'. At the bottom of the form area is a label 'Notification : ' followed by a large text area for input.

	Website	Link to Song
<input checked="" type="checkbox"/>	www.freesongs1.com	www.freesongs1.com/download1
<input checked="" type="checkbox"/>	www.freesongs2.com	www.freesongs2.com/download2
<input type="checkbox"/>	www.freesongs3.com	www.freesongs3.com/download3

Figure 7 : Results Tab

B. Results Tab

Under the Results Tab, the user is shown the list of sources which were found by the backend as copyright violations. A checkbox is displayed before every result so that the user can select the links to which he or she wants to send the notifications. This can easily be achieved by pressing the “Send Notification” button. A mail is drafted having the content specified in the notification text area and is then dispatched to the administrators of the selected websites. The sender's address is specified in a text box labeled as owner's email ID. All the links which are processed are deleted from this list and shifted to the list presented under processed results tab as shown in **Figure 8**.

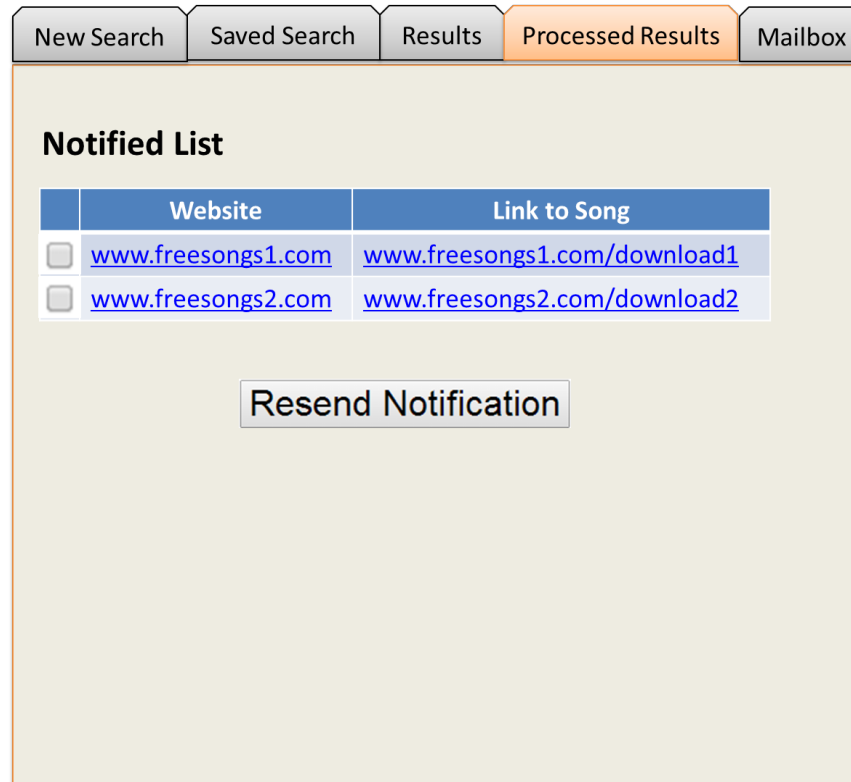


Figure 8 : Processed Results Tab

C. Processed Results Tab

In this tab, the user sees a table showing all the links of the websites, to which the notifications have been sent. If the user wishes to resend the notification to a particular website then he can select the corresponding website and can press the “Resend Notification” button. The basic purpose of this page is to separate out the processed results from the results that awaits processing (which are presented in the Results Tab).

D. Saved Search Tab

The user can revisit the inspected songs and view the related results (unprocessed) and processed by visiting the Saved Search Tab, shown in **Figure 9**. It has a list, ordered by the song names. By clicking the required song, its results are loaded and can be viewed under Results Tab and the Processed Results Tab.

E. Mailbox

This is a very useful tab in this system since it contains all the communication with the web administrators corresponding to particular song and website. It facilitates communication with the web administrators and/or content hosts. This tab is presented in **Figure 10**.

New Search	Saved Search	Results	Processed Results	Mailbox
------------	---------------------	---------	-------------------	---------

Songs List

A

[A Change Is Gonna Come](#) (Apollo Theatre)

[A Couple More Years](#) (*Hearts of Fire*)

[A Fool Such As I](#)

[A Hard Rain's A-Gonna Fall](#)

[A Satisfied Mind](#) (Philadelphia)

[Abandoned Love](#) ("Bitter end" version)

[Abraham, Martin And John](#)

D

[Dark As A Dungeon](#)

[Dark Eyes](#)

[Day Of The Locusts](#)

[Days Of '49](#)

[Dead Man, Dead Man](#)

[Dear Landlord](#)

Figure 9 : Saved Search Tab

New Search	Saved Search	Results	Processed Results	Mailbox
------------	--------------	---------	-------------------	----------------

Songs	Websites	Communication
<p>A</p> <p>A Change Is Gonna Come</p> <p>A Couple More Years</p> <p>A Fool Such As I</p> <p>A Hard Rain's A-Gonna Fall</p> <p>A Satisfied Mind</p> <p>Abandoned Love</p> <p>Abraham, Martin And John</p> <p>D</p> <p>Dark As A Dungeon</p> <p>Dark Eyes</p> <p>Day Of The Locusts</p> <p>Days Of '49</p> <p>Dead Man, Dead Man</p> <p>Dear Landlord</p>	<p>www.freesongs1.com</p> <p>www.freesongs2.com</p>	<p>Sender :</p> <div>Message1</div> <p>Website Admin :</p> <div>Message2</div> <p>New Message :</p> <div>Message3</div> <p><input type="button" value="Send"/></p>

Figure 10 : Mailbox

3.0 Defense

In this section, we would like to demonstrate the effectiveness of Music Inspector through the use of a concrete example. We refer back to some figures introduced earlier in **Section 2.3** to setup our example. Suppose we are the company owning copyrights for the Song “Dark Eyes” by Bob Dylan. We input the labels, whitelist, and mp3 file, as shown in **Figure 6**. Suppose that the whitelist file contains only two entries: mtv.com and muchmusic.com.

3.1 Crawler

In the crawler module, suppose that the additional terms we have are “torrent”, “download”, and “free”. We use these terms along with the the labels “Dark Eyes”, “Bob” Dylan”, and “Empire Emporium” to produce the search query:

(“Dark Eyes” or “Bob Dylan” or “Empire Emporium”) and (“torrent” or “download” or “free”)

For simplicity, suppose now that the crawler module queries 5 web services for potential copyright infringement, namely TorrentSite1, DownloadSite1, StreamSite1, SearchEngine1, and SearchEngine2. Moreover, suppose that we want at most 5 copyright infringements with at least a metric value of 11 reported. For the crawler search and filter algorithm, we want it to run at most 4 iterations and search 2 sources per iteration. Let us further suppose that when performing the search query, there a total of 8 search results. Statistics and metric calculations for each of these sources is shown in **Table 1**. The sources are then assessed in descending value of metric value.

Table 1: Metric Value Calculation for Potential Copyright Infringement Sources

Web Service	Location	Statistics	Metric Value
TorrentSite 1	Source 1: www.freesongs1.com	Seeders = 7 Leechers = 6	$7 + 6 = 13$
	Source 2: www.torrent.com/f2	Seeders = 5 Leechers = 6	$5 + 6 = 11$
DownloadSite 1	Source 3: www.freesongs2.com	numDownloads = 13	13
	Source 4: www.mtv.com/file4	numDownloads = 22	22
StreamSite 1	Source 5: www.freesongs3.com	numViews = 17	17
	Source 6: www.audiostream.com/f6	numViews = 4	4
SearchEngine 1	Source 7: www.searchengine1.com/f7	searchEngineRank = 4 searchEngineWt = 40	$40 * (\frac{1}{4}) = 10$

SearchEngine 2	Source 8: www.searchengine2.com/f8	searchEngineRank = 5 searchEngineWt = 60	$60 * (\frac{1}{5}) = 12$
----------------	------------------------------------	---	---------------------------

During the first iteration of the algorithm, Sources 4 and 5 are checked. Since Source 4 is whitelisted (mtv.com), it is not added to the list of infringements. On the other hand, Source 5 is added since its metric value is greater than 11. On the second iteration, Source 1 and Source 3 are added to the infringement list. On the third iteration, Source 2 and Source 8 are assessed to determine if the static threshold of 11 is reached. Note that despite the fact that Source 7 has a better rank than Source 8, it still receives a lower metric value than Source 8 because Source 7's searchEngineWt is lower than that of Source 8. Both Sources 2 and 8 are added to the list. Since there are now 5 sources in the list, the algorithm terminates. Moreover, if the algorithm did continue to check Source 6 and Source 7, these two sources are not added to the list, since their metric values are too low. The audio files hosted by Sources 1, 2, 4, and 8 would be directly pulled into our system. Since Source 5 is a stream, the system would first convert the stream to an audio file. Finally, 5 files originating from Sources 1, 2, 3, 5, and 8 are sent over to the acoustic verification module for validation.

3.2 Acoustic Verification

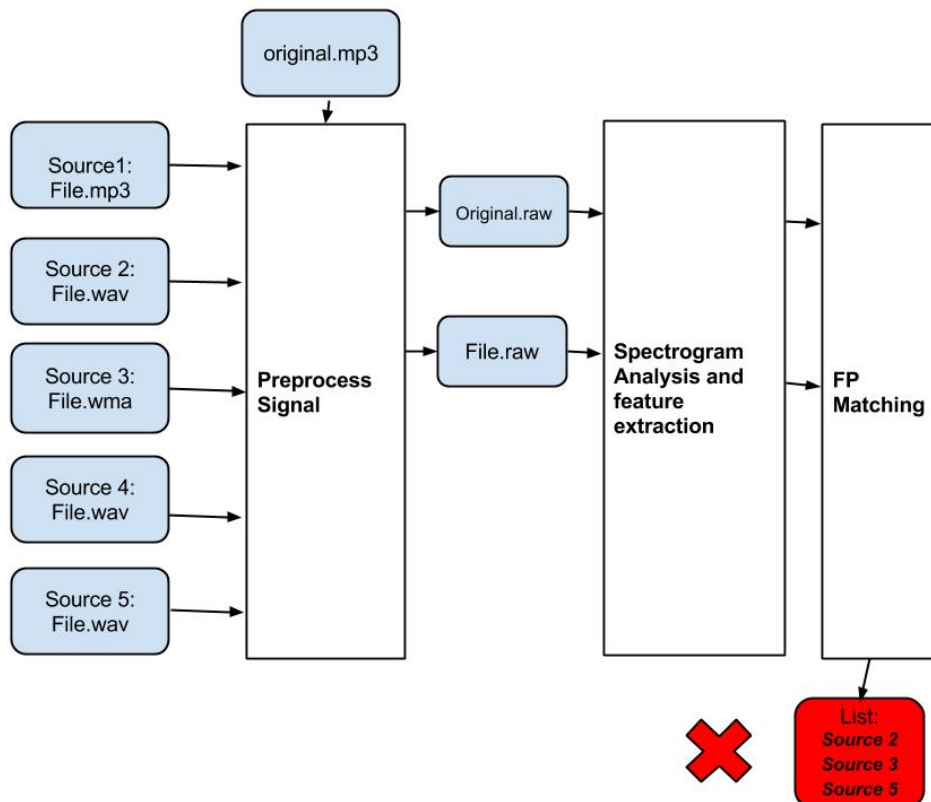


Figure 11 : Fingerprinting Example

The five sources obtained from web crawler are converted into a general format (in this case .raw). The original file is also converted into the “.raw” format. Both files are used to create spectrograms and fingerprints. Once the fingerprints are obtained, they are compared. If a match is found the source of the file is added to the “list” file. The fingerprint of the original copy is generated once and stored in the database. So everytime the fingerprint is only computed for the source file and later directly compared with the stored fingerprint of the original file.

3.3 Discussion

As seen from the example, our framework is able to produce a concise and manageable list of sources where music copyright violations have occurred. This is achieved via several design details. One detail is limiting the search space the crawler explore. Another is reporting files that have exceeded a threshold, so that the reported sources are significant sources of copyright infringement. Moreover, the acoustic verification module is not only capable of verifying entire songs that have been copied, but also segments of it, due to advance spectrogram analysis techniques. Lastly, but most importantly, Music Inspector enables copyright owners to check for copyright violations easily through an effective user interface. This user interface provides implementation that allows copyright owners to contact illegal content owners and hosts at the mere press of a button. Additionally, informational results for available review in an organized manner.

4.0 Limitations

Due to the need to keep our solution effective, we introduced several mechanisms to keep our system scalable in terms of searching and computation. This introduced some limitations to our system.

One limitation is that our system only produces a list of the top f infringements, as opposed to all infringements. That is to say, our system will not output sources where perhaps very few people can obtain copied music. The reason behind this limitation is that searching the Internet can lead to millions of hits, so it was necessary to limit the amount of processing that needed to be done. Another reason is that providing users with a list of every single infringement source that is not concise and manageable could overwhelm the user. This, in turn, can deter copyright users from using the system entirely.

Another limitation is that our system is not entirely self-contained. For example, if a torrent site was aware that our service was querying their site and reporting copyright violations, they may just simply block our service from accessing the site. Although this can be remediated by using dynamic IP addresses on a one-time basis, it may still pose some challenges for our Framework.

5.0 Recommendations and Future Work

There are two recommendations that we would like to make in order for Music Inspector to be

deployed and employed successfully.

In the crawler module, potential copyright infringement sources are assessed and then the content is downloaded for the acoustic verification module to verify. To prevent hosting copyright infringed content, we recommend that the files downloaded and streams converted be immediately discarded after the acoustic verification completes its verification.

Moreover, after receiving the results from the backend, the frontend provides the option of contacting content hosts, as seen in **Figure 10**. Contacting content uploaders, ISPs, and website administrators to remove stolen content from being downloaded has been a traditional approach. However, this approach is unscalable, as many individual users and administrators may have to be contacted. Furthermore, the solution is also temporary, as individuals continue finding new sources and content to illegal upload for distribution.

The music industry has been reluctant to change business models which they have greatly profited from in the pre digital era. This led to backlash from the Internet, as the industry lobbied to have bills such as the Stop Online Piracy Act to prosecute users for illegal downloading [3]. However, other industries selling content have been able to combat Internet piracy and produce profits. One example is the PC gaming industry, where game publisher Steam has been able to produce \$350 000 of profit per employee, a value greater than that of Google and Apple [2], through attractive deals and ease of use. Executives of the company have expressed that Internet piracy is a service problem [2]. Thus, this example may be a precedent for the music industry to follow.

We recommend that copyright owners not only use the results produced by Music Inspector as a list to contact people to take content down, but also as a learning source to improve current business models. For example, a music company may employ discriminatory pricing schemes to attract people to purchase their music.

In the future work, we can apply the system to identify copied videos. We can extract the audio of the movie file and perform the same check on that audio file. To acoustically verify long movie audio, we can extract important important fragments of the movie for audio comparison to reduce the time needed.

6.0 Conclusion

Although the Internet has brought new business opportunities, it has also changed the way in which consumers consume their content. Economically, the nature of Internet piracy has still not been mastered by the music industry. Hence, we propose Music Inspector, a technical solution that helps copyright owners track where on the Internet their work has been copied. Through the use of advance

algorithms, the framework presented is able to detect even small segments of music copied in a scalable manner. Ultimately, we hope that systems like Music Inspector are intermediary solutions to aid the music industry to re-tailor itself for business in the new digital era, so that music fans are able to compensate their artists for the creative work that they have produced.

References

- [1] Kazi Stastna, CBC News, "Copyright changes: how they'll affect users of digital content", 2011, <http://www.cbc.ca/news/canada/copyright-changes-how-they-ll-affect-users-of-digital-content-1.991691>.
- [2] Paul Tassi, "The Numbers Behind Steam's Success", 2/16/2012, <http://www.forbes.com/sites/insertcoin/2012/02/16/the-numbers-behind-steams-success>.
- [3] Jasmin Melvin, "SOPA and PIPA stopped after unprecedented protests from Wikipedia, reddit", 2012, <http://news.nationalpost.com/2012/01/20/sopa-stopped-after-unprecedented-online-protests/>.
- [4] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A Review of Algorithms for Audio Fingerprinting," in Workshop on Multimedia Signal Processing, 2002.
- [5] J. Haitsma, "A Highly Robust Audio Fingerprinting System," in Proceedings of ISMIR, 2002.
- [6] Chandrasekhar, M. Sharifi, and D. Ross, "Survey and Evaluation of Audio Fingerprinting Schemes for Mobile Query-By-Example Applications," in Proceedings of ISMIR, 2011.
- [7] A. Gionis, P. Indyk, R. Motwani, "Similarity Search in High Dimensions via Hashing," in Proceedings of VLDB, 1999.
- [8] V. A. Larsen. Combining audio fingerprints. Technical report, Norwegian University of Science and Technology, 2008. Available at <http://daim.idi.ntnu.no/>.