On the Use of Minhash and Locality Sensitive Hashing for Detecting Similar Lyrics

Francisco Javier Moreno Arboleda, Felipe Cortés Noreña, Benjamín Cruz Álvarez

***Abstract*—In this paper, we propose a retrieval system based on similarities between songs. We consider the similarity of songs regarding their lyrics, emotions, genres, or a combination of these attributes. To detect similar lyrics, we applied both minhash and locality-sensitive hashing (LSH) methods to a set of songs. We also applied the Watson Tone Analyzer service for detecting emotions. Although experiments with more songs are necessary, our results did not show, e.g., lyrics plagiarism. This finding suggests, at least from a textual point of view, that lyricists are careful on this matter. We also included some artificial similar songs in our set of songs to validate our proposal. Although there were false positives and true negatives, as expected in LSH, this experiment showed the fairness of our proposal.**

***Index Terms*— Music retrieval system, music plagiarism, similar lyrics, Jaccard index, locality-sensitive hashing, emotion mining**

I. Introduction

MUSIC plagiarism [1] is a very sensitive issue. A few identical fragments of lyrics from two songs can be enough to trigger a copyright lawsuit. For instance, in [2] are two fragments of two lyrics which gave rise to a lawsuit: "*I want it, I got it, I want it, I got it*." and "*You need it, I got it. You want it, I got it*." Similarly, in [3] two lyrics that include the same opening line "*I just took a DNA test, turns out I'm 100% that b*tch*" are mentioned. Another case is the song "*You Can't Catch Me*" by Chuck Berry which includes the fragment "*Here come a flat-top, he was moving up*" and the song "*Come Together*" by The Beatles which includes the fragment "*Here come old flat-top, he come groovin' up*".

In this paper, we apply both minhash and Locality-Sensitive Hashing (LSH) methods to detect similar lyrics, where two lyrics are considered similar if they share at least a few fragments of the same words. To narrow down the search for plagiarisms, the analyst can specify additional conditions (filters) focused on the musical genres and emotions of the songs. Accordingly, the problem can be described as follows.

We propose a song retrieval system based on similarity between songs according to a) their lyrics (plagiarism detection based on identical textual fragments), through both minhash and LSH, b) their emotions, and c) other attributes, such as their artists and their musical genres. Thus, queries could be formulated to find pairs of songs considering, e.g., their similarity according to their lyrics, their emotions, their genres, or a combination of these and other attributes. For example, finding pairs of songs (s_1, s_2) considering their similarity according to their lyrics and that also:

1) Share the same set of genres, e.g., {"Rock"}.
2) Belong to different genres, e.g., s_1 belongs to "Pop" and s_2 to "Metal".
3) Share the same set of emotions, e.g., {"Sadness", "Fear"}.
4) Are analogous to 2 but for emotions.
5) Are a combination of 1 and 3, 1 and 4, 2 and 3, or 2 and 4.

Among others.

Our system is outlined in Figure 1.

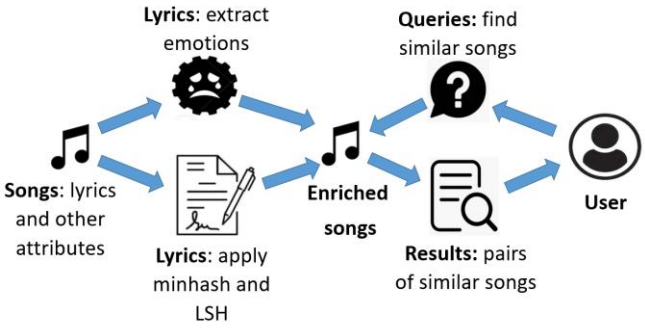

Fig. 1. Outline of our song retrieval system

II. Definitions

Let S = {s_1, s_2, ..., s_g} be a set of songs, where a song is a 7-tuple (s_{id}, s_{title}, s_{artist}, s_{genre}, $s_{emotion}$, s_{lyrics}, $s_{signature}$), where

- s_{id}, a positive integer, is the unique identifier of the song.
- s_{title}, a string, is the title of the song.
- s_{artist} = {a_1, a_2, …, a_t} is a set of the names (strings) of the artists who perform the song; usually, s_{artist} will be a singleton; however, we consider music collaborations, e.g., duets.
- s_{genre} = {g_1, g_2, …, g_e} is a set of the names (strings) of the musical genres in which the song has been classified; usually, s_{genre} will be a singleton, i.e., the musical genre of the song; however, we consider

Manuscript received February 15, 2021; revised January 17, 2022.

Francisco Javier Moreno Arboleda is an associate professor at the Departamento de Ciencias de la Computación y de la Decisión, Universidad Nacional de Colombia, Sede Medellín, Colombia (phone: 604-425-5376; e-mail: fjmoreno@unal.edu.co).

Felipe Cortés Noreña is a computer science engineer at Universidad Nacional de Colombia, Sede Medellín; fcortesn@unal.edu.co.

Benjamín Cruz Álvarez is a computer science engineer at Universidad Nacional de Colombia, Sede Medellín; becruza@unal.edu.co.

that a song may be classified in several genres. In addition, when the musical genre of a song is unknown or unclear, we use a special value "*Unknown*".

- $s_{emotion}$ = {e_1, e_2, …, e_m} is a set of the emotions (strings) of the song generated from its lyrics. $s_{emotion}$ can be obtained from a emotions analysis tool, e.g., the Watson Tone Analyzer (WTA) [4]. We present a detailed example of how to obtain $s_{emotion}$ from the lyrics of a song in Section III.
- s_{lyrics}, a string, is the lyrics of the song.
- $s_{signature}$ is the signature of s_{lyrics}. $s_{signature}$ is obtained through minhash [5] and is a sequence of non-negative integers [i_1, i_2, ..., i_n] where n is the number of hash functions.

Note that two songs, s_i and s_j, could be equal in all their attributes, except in their s_{id}. This case can happen, e.g., when an artist performs the same song in different versions (e.g., live version and studio version) and maintains the same genre and the same lyrics (which implies the signatures being the same, i.e., there is a functional dependency: $s_{lyrics} \rightarrow s_{signature}$). However, the analyst might include some constraints on S, e.g., to avoid two songs with the same lyrics and the same artist.

Example 1. Let S = {s_1, s_2, s_3, s_4, s_5} where:

- s_1 = (1, "*Club Tropicana*", {"Wham!"}, {"Pop"}, {"Joy"}, {"*Let me take you to the place… But don't worry, you can suntan!*"}, [2, 1, 2, …]). Lyrics taken from [6].
- s_2 = (2, "*Loverboy*", {"Mariah Carey", "Da Brat", "Ludacris", "Shawnnta"}, {"Pop", "Rap"}, {"Joy", "Confident"}, {*"(Ah) my girl… A loverboy for me"*}, [1, 3, 0, …]). Lyrics taken from [7].

And so on.

A. Obtaining the signature from the lyrics: minhash

The Jaccard similarity coefficient (JSC) or Jaccard index [5] measures the similarity between two sets. Let sets A and B, then the JSC is |A ∩ B|/|A ∪ B|.

Example 2. Let A = {0, 2, 4, 7, 9, 10, 11, 15} and B = {2, 7, 9, 13, 15, 20} then |A ∩ B| = 4, |A ∪ B| = 10 and JSC = 4/10.

On the other hand, minhash [5] is an efficient method to estimate this coefficient. Basically, each set is condensed into a *signature*. Then, the signatures are used to estimate the JSC. In our case, we are interested in finding the similarity of lyrics. Thus, the lyrics of a song are treated as a set of text fragments (called *shingles*). A *k*-shingle (also called *k*-gram) is composed of *k* contiguous subsequences of tokens within a document. A token can be, e.g., a character or a word. Consider the following example. Here, we consider the following very short lyrics:

Lyrics song 1 (LS_1): "*I love you and your smile*".
Lyrics song 2 (LS_2): "*I love your smile*".

For this example, we will use 2-word shingles. Thus, the set of 2-shingles of the lyrics of a song is all the possible consecutive pairs of two words. For LS_1 we have {"*I love*", "*love you*", "*you and*", "*and your*", "*your smile*"} and for LS_2 {"*I love*", "*love your*", "*your smile*"}. The JSC for these two sets is 2/6 = 1/3.

Next, we explain how to obtain the signature of the lyrics of a song. The two previous lyrics generate the set of shingles: {"*I love*", "*love you*", "*you and*", "*and your*", "*your smile*", "*love your*"}. Each shingle is transformed (mapped) into an integer (a bucket number) using a hash function. For this example, we will assume the mapping of Table I.

TABLE I
CORRESPONDENCE BETWEEN SHINGLES AND THEIR NUMBERS

2-shingle	"*I love*"	"*love you*"	"*you and*"	"*and your*"	"*your smile*"	"*love your*"
Shingle number	1	2	3	4	5	6

The next step is constructing a matrix that indicates which shingles the lyrics of each song have (1 indicates that it has it and 0 that it does not); see Table II.

TABLE II
MATRIX OF LYRICS AND SHINGLE NUMBERS

Shingle number	LS_1	LS_2
1	1	1
2	1	0
3	1	0
4	1	0
5	1	1
6	0	1

Now, we construct the following matrix: we define n hash functions each of the form:

$$h(x) = (ax + b) \bmod c$$

Where x is the shingle number and c is the next prime number greater than the total number of shingles (here, c = 7 because the total number of shingles is 6). For more details on the definition of these functions and the conditions they must meet (e.g., they must be linearly independent) see [5]. For this example, we define four functions (n = 4):

Hash function 1: $hf_1(x) = (1x + 4) \bmod 7$.
Hash function 2: $hf_2(x) = (2x + 3) \bmod 7$.
Hash function 3: $hf_3(x) = (6x + 5) \bmod 7$.
Hash function 4: $hf_4(x) = (3x + 1) \bmod 7$.

Thus, the signature will be a sequence of 4 non-negative integers. In Table III we show the results of applying these four hash functions to the shingle numbers from Table II.

TABLE III
RESULTS OF APPLYING THE FOUR HASH FUNCTIONS. THE VALUES OF THE SIGNATURE FOR LS_2 = **[2, 1, 0, 2]** ARE SHOWN IN BLUE

Shingle number (x)	LS_1	LS_2	$hf_1(x)$	$hf_2(x)$	$hf_3(x)$	$hf_4(x)$
1	1	1	5	5	4	4
2	1	0	6	0	3	0
3	1	0	0	2	2	3
4	1	0	1	4	1	6
5	1	1	2	6	0	2
6	0	1	3	1	6	5

Finally, we obtain the signature (attribute $s_{signature}$) of LS_1 and LS_2 from Table III as follows. For example, for LS_2:

- We need to find in column $hf_1(x)$ the smallest number such that column LS_2 (i.e., the third column of Table III) has the value 1 in the same row. In Table III, this number is **2**.
- We do the same with $hf_2(x)$, $hf_3(x)$, and $hf_4(x)$ columns, and we get 1, 0, and 2, respectively.

Thus, the signature for LS_2 is [**2**, **1**, **0**, **2**] (see the blue numbers in Table III). Following the same process for LS_1, we construct the signature [0, 0, 0, 0]. Then, with these signatures, we create a signature matrix; see Table IV.

TABLE IV
SIGNATURE MATRIX

LS_1	LS_2
0	2
0	1
0	0
0	2

To estimate the JSC from the signatures, the rows of Table IV in which the signatures are equal are counted and divided by the total elements of the signature (i.e., n). For this example, the signatures only match at one position (the third row in Table IV, the green numbers); therefore, the JSC = 1/4. Note that in this example, this value was not equal to the one obtained directly from the two sets of shingles; this inequality is not an error, since the method generates an estimation of the JSC. Depending on i) the size of the documents, ii) their type (emails, songs, scientific papers, etc.), and iii) the level of accuracy desired in the coefficient estimation; the analyst must define the size of the shingles and the number of hash functions (n) for generating the signature; some ideas about it are discussed in [5].

On the other hand, LSH is a method for reducing the number of comparisons (when two signatures are compared) in the signature matrix. The method divides a matrix of n rows in b bands, each band has r rows ($b * r = n$). Thus, n is the signature length and r the band size.

For example, consider the matrix of three signatures LS_1, LS_2, and LS_3 from Table V where $n = 4$. In this example, the matrix is divided into 2 bands ($b = 2$) of 2 rows ($r = 2$) each.

TABLE V
SIGNATURE TABLE

LS_1	LS_2	LS_3	
0	2	0	Band 1
0	1	0	
0	0	0	Band 2
0	2	2	

The next step is analyzing the bands: if two signatures are equal in *at least* one band, they are considered a *possibly* similar pair of documents. Thus, when analyzing band 1 from our example, we observe that signatures LS_1 and LS_3 are identical (0, 0 and 0, 0); therefore, the pair (LS_1, LS_3) is a candidate pair. The same occurs in band 2 for LS_2 and LS_3 (0, 2 and 0, 2). Finally, the signatures are used to estimate the JSC for the candidate pairs.

B. Obtaining the emotions from the lyrics: Watson Tone Analyzer

For obtaining the emotions (attribute $s_{emotion}$) from the lyrics, we apply the WTA [4]. Seven emotions are considered in this tool: Joy, Anger, Sadness, Fear, Confident, Analytical, and Tentative. The tool detects the emotions from the entire document (*document-level*) and from each sentence (*sentence-level*). For each emotion, the WTA generates a score between 0 and 1. Only those emotions with scores greater than or equal to 0.5 are included in the result. This result means that the document (or sentence) is characterized by these emotions. A document (or a sentence) is characterized by zero, one, or more emotions (each with a score greater than or equal to 0.5). In our proposal, we detect the emotions from the s_{lyrics} at the *document-level*, i.e., from the entire lyrics.

Example 3. For the lyrics of the song 1 ("*I love you and your smile*"), the document-level emotion given by the WTA was {"Joy"} with score 0.98. For the lyrics of the song 2 ("*I love your smile*"), the document-level emotion given by the WTA was {"Joy"} with score 0.99.

III. Putting All the Pieces Together: An Algorithm for Finding Similar Songs

The life cycle of our proposal begins with obtaining the values for attributes $s_{signature}$ and $s_{emotion}$ from the lyrics. We called this task *song enrichment*.

A. Lyrics normalization

The lyrics could be submitted to a normalization process to standardize the language; however, this is a complex process. For example, suppose the lyrics of two songs include the following fragments:

Lyrics song 1: "…*I don't love you*…".
Lyrics song 2: "…*I donut luv ya*…".

These two fragments represent the same idea: "*I do not love you*"; however, converting "*I donut luv ya*" in "*I do not love you*" is not a trifling task since such a conversion would involve an analysis of deviations from standardized English. Thus, usual deviations in lyrics should be considered such as incorrect conjugations ("*I be*" instead of "*I am*", "*she don't*" instead of "*she doesn't*"), contractions (e.g., "*wanna*", "*shoulda*", "*kinda*", "*dunno*" for "*don't know*"), misspellings (e.g., "*luv*" for "*love*", "*wot*" for "*what*"), phonetic character replacements (e.g., "*sk8er boi*"), interjections ("*Ah!*", "*Ohhh*"), repetitions of verses or sentences (some lyrics sites use, e.g., "X *p*", to show that a sentence or a verse repeats *p* times), acronyms (e.g., "*The G.O.A.T.*" stands for "*The Greatest Of All Time*"), among many others [8].

Furthermore, from a statistical point of view such a task may be non-significant, i.e., the emotions associated with the lyrics might not change if such normalization is done, except in the edge cases when the lyrics are very short and such standardization would be decisive (e.g., suppose the lyrics of a song are only "*I'm o'erjoyed*"; as a consequence, the WTA does not understand the word "*o'erjoyed*" and does not report

emotions. If "*o'erjoyed*" were standardized to "*overjoyed*", the tool would report the emotion Joy). Due to the above, to detect the emotions we only removed markers that usually appear in square brackets in some lyrics (e.g., "[*CHORUS*]" or when it is indicated that an artist sings a fragment of a song, e.g., "[*C-Murder*] … *I'ma rida*" [9]).

Regarding the signature generation, the non-standardization of the lyrics will cause that some shingles between two lyrics are considered different (e.g., "*luv ya*" and "*love you*"), although statistically these omissions are non-significant (except for very short lyrics, e.g., with one or two sentences, it could affect plagiarism identification). On the other hand, to construct the signature, we only i) removed the punctuation marks, ii) replaced newlines with spaces, and iii) converted the lyrics to lowercase.

B. *Song enrichment and queries*

First, we define the size of the shingles (i.e., the number of words that make up the shingle) and the number (n) of hash functions (which defined the length of the signature). We will use these values with all the songs. Then, we detect the emotions of a song using the WTA. For this step, we apply the WTA to the lyrics after first removing the mentioned markers, detecting the emotions at the document-level. Next, we perform the basic normalization process mentioned at the end of subsection IIIA and we construct the signature using minhash. We show this process in the two following functions in Figure 2.

Once we have the set of enriched songs S, we can return the results based on the user's queries. Assuming we have our set S, e.g., in a database table called S, where attributes sartist, sgenre, semotion, and ssignature are arrays (collections), we could formulate SQL-like queries. Consider, e.g., the query "Find the pairs of rock songs (s1, s2) that are possibly similar based on their lyrics". We could formulate the following SQL-like query, see Figure 3.

```
 1: Function songsEnrichment(S)
 2: Input
 3: S: A set of songs to be enriched.
 4: Output:
 5: S: A set of enriched songs.
 6: Begin:
 7: Set shingleSize       /* Size of the shingles */
 8: Set n                 /* Number of hash functions */
 9: Foreach song ϵ S Do
10:   /* Call Enrichment() function, see next funtion */
11:   S.song = Enrichment(song, shingleSize, n)
12: End Foreach
13: Return S;
14: End songsEnrichment
```

```
 1: Function Enrichment(song, shingleSize, n)
 2: Input
 3: song: A song with empty attributes semotion and ssignature.
 4: shingleSize: Size of the shingles.
 5: n: Number of hash functions.
 6: Output:
 7: song: The song with attributes semotion and ssignature filled.
 8: Begin
 9: /* Lyrics normalization, line 11 */
10: /* Remove markers such as [...] from lyrics */
11: song.slyrics = removeMarkers(song.slyrics);
12: /* Detect emotions */
13: song.semotion = WatsonToneAnalizer(song.slyrics);
14: /* Continue lyrics normalization, lines 16, 18, and 20 */
15: /* Remove punctuation marks */
16: song.slyrics = removePunctuationMarks(song.slyrics);
17: /* Replace newlines with spaces */
18: song.slyrics = replaceNewLinesWithSpaces(song.slyrics);
19: /* Convert to lowercase */
20: song.slyrics = lowercase(song.slyrics);
21: /* Construct signature */
22: song.ssignature = minHash(shingleSize, n, song.slyrics);
23: Return song; /* Return enriched song
24: End Enrichment
```

Fig. 2. songsEnrichment and Enrichment functions

```
SELECT s1.sid, s1.stitle, s2.sid, s2.stitle
FROM S AS s1, S AS s2
WHERE 'Rock' MEMBER OF s1.sgenre AND
      'Rock' MEMBER OF s2.sgenre AND
      s1.sid < s2.sid AND
      LSH(s1.ssignature, s2.ssignature, :bandsize) IS TRUE;
```

Fig. 3. Query to find pairs of rock songs possibly similar based on their lyrics

```
SELECT s1.sid, s1.stitle, s2.sid, s2.stitle
FROM S AS s1, S AS s2
WHERE 'Pop' MEMBER OF s1.sgenre AND
      'Joy' MEMBER OF s1.semotion AND
      'Metal' MEMBER OF s2.sgenre AND
      'Sadness' MEMBER OF s2.semotion AND
      'Anger' MEMBER OF s2.semotion AND
      s1.sid < s2.sid AND
      LSH(s1.ssignature, s2.ssignature, :bandsize) IS TRUE;
```

Fig. 4. Query to find pairs of songs (s_1, s_2) possibly similar lyrics where s_1 is joy and pop and s_2 is sad, anger, and metal

Here, we use the MEMBER OF [10] operator to test membership of an element in an array (collection). *:bandsize* is a parameter. In a similar way, the query "*Find the pairs of songs (s_1, s_2) that are possibly similar based on their lyrics where s_1 is joyful pop and s_2 is sad, anger, and metal*" could be formulated as we show in Figure 4.

IV. Experiments

As we pointed out in Section II, we used the WTA [4] to detect the emotions of a song from the lyrics. We considered eight musical genres [11]: Pop, Rock, Rap, Country, Blues, R&B, Metal, and Electronic. For the sake of simplicity, we did not consider musical subgenres. We took the lyrics from [9] and considered a set S of 541 songs. The lyrics of each song were normalized as we showed in Figure 2. The average number of changes that the lyrics of a song underwent was the following: 2 markers and 35 punctuation marks were removed, 72 newlines were replaced with spaces, and 84 uppercase letters were converted to lowercase.

Next, we proceeded to identify the pairs of songs possibly similar according to their lyrics (initially, without considering emotions or other attributes). For this process, we defined three parameters: shingle size, band size, and the number of hash functions. We define a *variation* as a triplet of values (shingle size, band size, and number of hash functions). For each variation, we got a set of matches, where a match is a candidate pair of songs (s_1, s_2) possibly similar (based on their lyrics). Note that the pair (s_1, s_2) is considered equal to the pair (s_2, s_1).

Each parameter was varied as follows:

- **Shingle size:** we considered values 1, 2, 3, and 4 consecutive *words*. We did not consider greater values because the matches tended to zero. This result makes sense because the longer the shingles are (i.e., longer sequences of consecutive words), the less likely that there will be shingles in common between two lyrics. For instance, let there be two lyrics: "*I love you*" and "*I love her*", if we define the shingles of size 2, these songs have a shingle in common ("*I love*"); but if we define the shingles of size 3, these songs do not have shingles in common.

- **Band size (*r*):** we considered values 2, 4, 8, 16, and 32. For values 8, 16 and 32 the matches tended to zero. This result makes sense because the greater the band size, the lower the probability of two signatures being identical in any band. Thus, in the example in Section II, Table V, if instead of using bands of size 2, we use bands of size 1, LS_1 and LS_2 would have an identical band (the third one, with zero values). With bands of size 2, LS_1 and LS_2 do not have identical bands.

- **Number of hash functions (*n*):** we considered values 16, 32, 64, 128, 256, 512, 1024, and 2048. We observed that the greater the number of hash functions, the greater the number of matches. This result makes sense because the greater the number of hash functions, the longer the signature, then there would be more bands, and the greater the probability of two signatures being identical in at least one band. This behavior was confirmed in our experiments (at least until 2048 hash functions). Indeed, if the number of hash functions tends to infinity, the number of matches will tend to the maximum number of matches: $w*(w-1)/2$, where w is the number of songs (in our experiments: 541*540/2 = 146070).

Figures 6, 7, 8 and 9 show the results for variations with shingle size 1, 2, 3, and 4, respectively. Indeed, note that the greater the number of hash functions and the smaller the band size, the greater the number of matches. Figure 9-a and Figure 9-c show this trend (we did not show the other results because for the other variations, the number of matches tended to zero). In addition, note that, as the shingle size increases, the number of matches decreases. Figure 5 also shows that when band size = 2, as the number of hash functions increases, the number of matches stabilized in as many matches as possible.

We tested the following queries with the following parameters: shingle size = 3, number of hash functions =

1024, and band size = 2. The queries were the following: find the pairs of songs (s_1, s_2) that are possibly similar based on their lyrics and that additionally: 1. Both are rock songs, 2. s_1 is metal and s_2 is pop, 3. Both are joyful (joy), 4. s_1 is tentative and s_2 is anger, and 5. s_1 is joyful pop and s_2 is sad metal.

We performed the experiments on an Intel core i-5 2.20 GHz processor, with 8 GB memory with Windows 10 Pro 64 bits. We show the results in Table VI. With these hardware specifications, the average time of the Enrichment function was 2497 ms per song, i.e., a total of 1351.36 s for the 541 songs. We show the time results for specific queries in Table VI.

Shingle Size = 1	Band Size				
Hash Functions	2	4	8	16	32
16	16982	209	1	1	1
32	23183	308	3	2	1
64	41946	576	8	2	1
128	77312	837	3	2	1
256	123692	4446	9	2	2
512	133911	5466	5	2	2
1024	137408	9786	8	3	2
2048	140785	18181	11	3	2

Fig. 5. Results for variations with shingle size = 1

Shingle Size = 2	Band Size				
Hash Functions	2	4	8	16	32
16	80	0	0	0	0
32	116	0	0	0	0
64	631	0	0	0	0
128	1072	0	0	0	0
256	2610	3	0	0	0
512	5168	4	0	0	0
1024	7118	3	0	0	0
2048	8827	4	0	0	0

Fig. 6. Results for variations with shingle size = 2

Shingle Size = 3	Band Size				
Hash Functions	2	4	8	16	32
16	2	0	0	0	0
32	2	0	0	0	0
64	8	0	0	0	0
128	23	0	0	0	0
256	18	1	0	0	0
512	39	1	0	0	0
1024	118	1	0	0	0
2048	202	1	0	0	0

Fig. 7. Results for variations with shingle size = 3

Shingle Size = 4	Band Size				
Hash Functions	2	4	8	16	32
16	1	0	0	0	0
32	1	1	0	0	0
64	1	0	0	0	0
128	4	0	0	0	0
256	7	0	0	0	0
512	13	0	0	0	0
1024	9	1	0	0	0
2048	34	2	0	0	0

Fig. 8. Results for variations with shingle size = 4

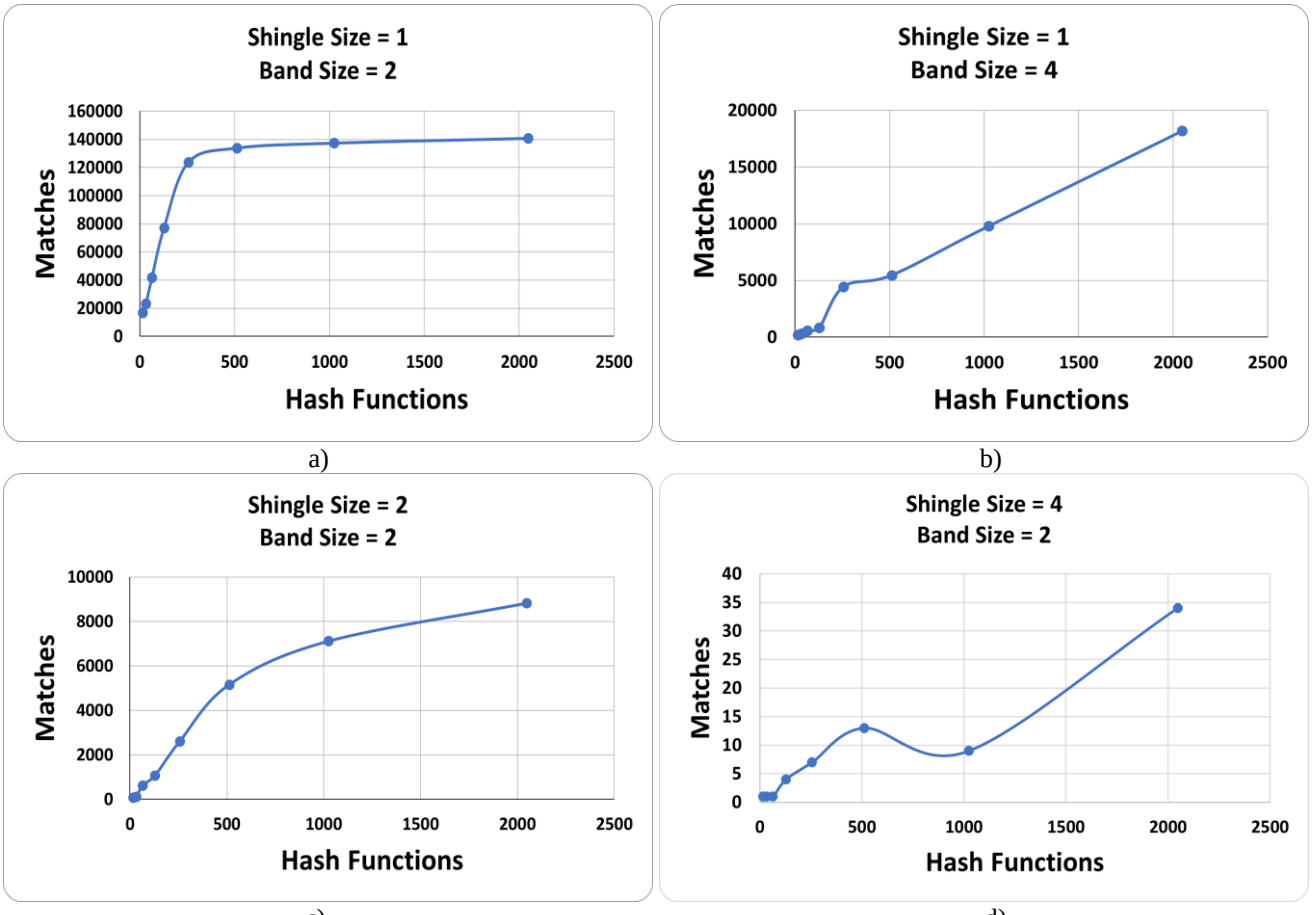

Fig. 9. Results for variations: a) shingle size = 1 and band size = 2; b) shingle size = 1 and band size = 4; c) shingle size = 2 and band size = 2; d) shingle size = 4 and band size = 2

TABLE VI
RESULTS OF THE EXPERIMENTS FOR THE QUERIES WITH BAND SIZE 2

Query	Number of matches	Runtime (ms)
1	24	769.846
2	5	619.487
3	54	907.656
4	8	810.618
5	2	332.411

Note that the number of matches is high considering that the queries are quite specific. However, the band size = 2 causes the method (LSH) to generate many pairs of songs as candidates, although there may be many false positives. This behaviour was confirmed when selecting and analyzing some matches (we selected pairs of songs that were not covers): the shingles in common were so few (one or two) and irrelevant as to suggest plagiarisms. For instance, the match "*Mirror's Reflection*" by Taproot and "*A Conspiracy*" by The Black Crowes, only shares the shingle "*what you don't*" and has a JSC of 0.00390625; the match "*You Never Cry Like a Lover*" by Eagles and "*Manchester England*" by Hair, only shares the shingle "*i believe in*" and has a JSC of 0.00440529; and finally, the match "*The End of Pain*" by Candlemass and "*Somewhere Down the Road*" by Faith Hill, only shares the shingles "*i see the*" and "*the end of*" and has a JSC of 0.008403361.

When we increased the band size to 4 we got zero matches for the five queries. This result suggests that for the queries, no pairs of possible similar songs were identified (i.e., possible plagiarisms were not identified).

On the other hand, by dispensing with the genre and emotions filters, i.e., finding the pairs of songs (s_1, s_2) that are possibly similar based only on their lyrics, with the initial parameters, we got two matches with JSC > 0.25, but both turned out to be covers. In the appendix, in Table AI, we show the lyrics of one of these matches ("*Thank You*" by Tori Amos and "*Thank You*" by Led Zeppelin). There, we also show their shingles in common (49 in total); see Table AII. This match had a JSC of 0.29614321. Furthermore, the match "*Hurt*" By Johnny Cash and "*Hurt*" By Nine Inch Nails shares 117 shingles and has a JSC of 0.85401459.

Next, we did a comparison with Turnitin (*https://www.turnitin.com*) to find similar songs to "*Thank You*" by Led Zeppelin. However, Turnitin *does not provide search filters* (e.g., by songs attributes or by songs emotions). In addition, Turnitin compares a specific song against a large collection of possible matches *of all kind of documents* (not necessarily lyrics); this is a big difference with our proposal, where we find pairs of similar songs from a set of (enriched) songs. In Figure 10 we show the Turnitin results for this song. We only got matches *with the same song*, but from various internet sources. Some of the matches were unavailable on the internet (broken links), some matched only one verse of the song.

We also did the same experiment with DiffChecker (*https://www.diffchecker.com*). DiffChecker does not provide *search filters* either; and its interface only allows two songs to be compare at a time. Here, we compared "*Thank You*" by Tori Amos and "*Thank You*" by Led Zeppelin. The results are presented in Figure 11. Note that DiffChecker highlights the text fragments that *differ* between the documents to be compared.

To verify that the method does not only detect covers, we conducted the following experiments. We chose a subset of arbitrary songs from our set S of 541 songs, and from each one we extracted a verse of their lyrics. With these verses, we created an artificial song (a "Frankenstein" song) which was included in the set S.

For the first experiment, we extracted verses from eight songs: "*(At Your Best) You Are Love*" by The Isley Brothers, "*A Home*" by Dixie Chicks, "*Faster Harder Scooter*" by Scooter, "*All I Could Bleed*" by Testament, "*100 Dollar Bag*" by Beenie Man, "*3rd Ward Solja*" by Juvenile, "*(You Make Me Feel Like) a Natural Man*" by Rod Stewart, and "*(I've Had) the Time of My Life*" by Bill Medley and Jennifer Warnes (see appendix, Table AIII).

In Table VII, we show the matches for the artificial song.

The method returned 7 of the 8 songs as matches. In addition, the method returned another 3 matches (3 songs that had no verses in common with the artificial one). Thus, the precision was 0.7 (precision = true positives / (true positives + false positives)).

This result is because LSH is probabilistic in that it can generate false positives, as stated in [5]: "*There will also be false positives – candidate pairs that are evaluated, but are found not to be sufficiently similar*". In addition, it is also stated that "*Choose a threshold t that defines how similar documents have to be in order for them to be regarded as a desired 'similar pair'. Pick a number of bands b and a number of rows r such that b * r = n, and the threshold t is approximately (1/b)1/r. If avoidance of false negatives is important, you may wish to select b and r to produce a threshold lower than t"* [5],

Thus, for reducing the false positives, we increased the size band to 4. In Table VIII we show the matches.

Indeed, this action caused the false positives to be removed, but *the true positives were also reduced* (3 of the 7 matches were removed from Table VII). Thus, size band = 2 offered a tradeoff between false and true positives.

In a second experiment, we created another artificial song with verses of joyful songs. We extracted verses from four songs: "*Fight*" by Amy Grant, "*#1 With a Bullet*" by Lindsay Pagano, "*Steal Away (The Night)*" by Ozzy Osbourne, and "*All Your Love*" by Eric Clapton; see appendix, Table AIV.

In Table IX, we show the matches for the artificial song, but this time we only consider *joyful* songs.

The method returned the 4 songs as matches, i.e., precision = 1 and there were no false positives. Next, we increased the size band to 4 and 2 of the 4 matches were removed.

Next, we performed a basic experiment with synonyms. We consider the list of synonyms from *https://www.hitbullseye.com/Vocab/List-of-Synonyms.php*. There, they present a table: in the first column, called simply *Word*, there is a word and in subsequent columns, there are four synonyms of the word. These columns are called *Synonym-1*, *Synonym-2*, *Synonym-3*, and *Synonym-4*. We show a fragment of this table in Table X.

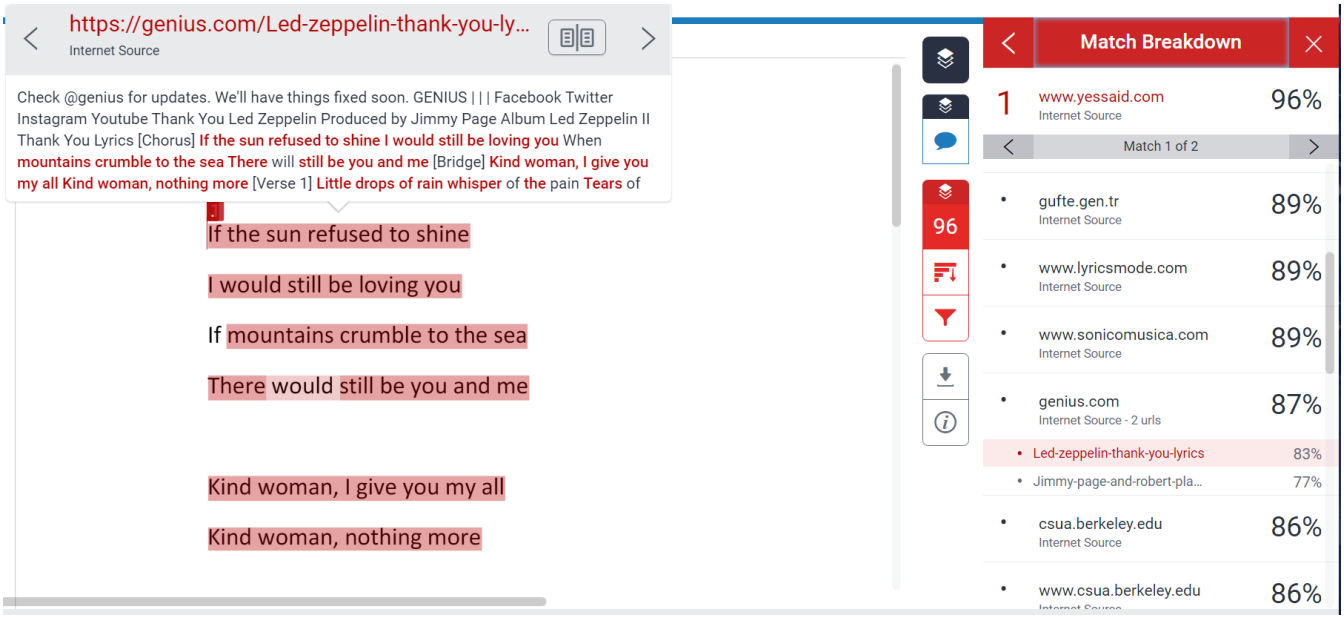

Fig. 10. Turnitin results for the song "*Thank you*" by Led Zeppelin

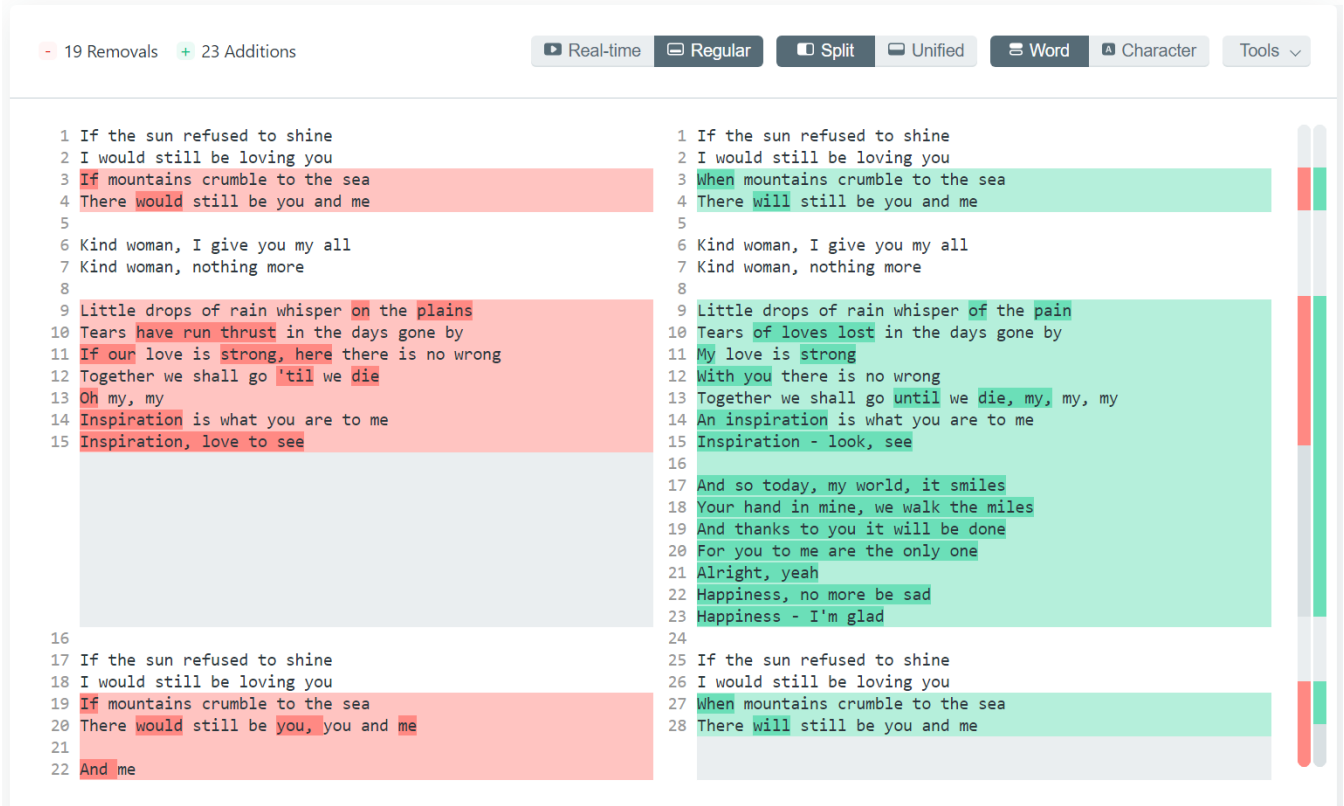

Fig. 11. DiffChecker results for the songs "*Thank You*" by Tori Amos and "*Thank You*" by Led Zeppelin

TABLE VII
MATCHES FOR THE ARTIFICIAL SONG WITH BAND SIZE = 2

Song	Jaccard Index	Shingles in common
"*3rd Ward Solja*" by Juvenile	0.17689530685920576	147
"*(At Your Best) You Are Love*" by The Isley Brothers	0.05504587155963303	36
"*Faster Harder Scooter*" by Scooter	0.1610738255033557	96
"*100 Dollar Bag*" by Beenie Man	0.034653465346534656	28
"*All I Could Bleed*" by Testament	0.08159722222222222	47
"*A Home*" by Dixie Chicks	0.07272727272727272	44
"*(You Make Me Feel Like) a Natural Man*" by Rod Stewart	0.09698996655518395	58
"*I'll Be Your Everything*" by Youngstown	0.004601226993865031	3
"*One Way Ticket (Because I Can)*" by LeAnn Rimes	0.0013774104683195593	1
"*Ti Amo*" by Gina G	0.001557632398753894	1

TABLE VIIII
MATCHES FOR THE ARTIFICIAL SONG WITH BAND SIZE = 4

Song	Jaccard Index	Shingles in common
"*3rd Ward Solja*" by Juvenile	0.17689530685920576	147
"*Faster Harder Scooter*" by Scooter	0.1610738255033557	96
"*A Home*" by Dixie Chicks	0.07272727272727272	44
"*(You Make Me Feel Like) a Natural Man*" by Rod Stewart	0.09698996655518395	58

TABLE IX
MATCHES FOR THE JOYFUL ARTIFICIAL SONG WITH BAND SIZE = 2

Song	Jaccard Index	Shingles in common
"*Fight*" by Amy Grant	0.15079365079365079	57
"*Steal Away (The Night)*" by Ozzy Osbourne	0.15666666666666668	47
"*#1 With a Bullet*" by Lindsay Pagano	0.21140939597315436	63
"*All Your Love*" by Eric Clapton	0.1952191235059761	49

TABLE X
FRAGMENT OF TABLE OF SYNONYMS

Word	Synonym-1	Synonym-2	Synonym-3	Synonym-4
Anger	Enrage	Infuriate	Arouse	Nettle
Come	Approach	Advance	Near	Arrive
Have	Acquire	Gain	Maintain	Believe
Love	Like	Admire	Esteem	Fancy
Run	Race	Sprint	Dash	Rush
Stop	Cease	Halt	Stay	Pause
Tell	Disclose	Reveal	Show	Expose

We chose an arbitrary song from our set S of 541 songs. The song was "*Stay*" by The Kid LAROI and Justin Bieber. In our previous experiments, this song got zero matches. We took the lyrics of this song and took each of their words. For each word *wd* of the song, we checked if *wd* was equal to Synonym-1, Synonym-2, Synonym-3, or Synonym-4; if true, then we replaced *wd* with the corresponding word in the first column.

In Table XI we show one verse of the song. On the left, we show the original verse and on the right, we show the verse with the replacements (synonyms, see words underlined in the table). The total number of distinct replacements for the entire song was three (the third replacement was in another verse, in this sentence: "*I feel **like** you can't feel*" where '***like***', an adverb, was replaced by '***love***', a verb, i.e., "*I feel **love** you can't feel*").

TABLE XI
ORIGINAL VERSE AND MODIFIED VERSE

Original verse	Modified verse
When I'm away from you, I miss your touch (ooh-ooh) You're the reason I **believe** in love It's been difficult for me to trust (ooh-ooh) And I'm afraid that I'ma fuck it up Ain't no way that I can leave you stranded 'Cause you ain't ever left me empty-handed And you know that I know that I can't live without you So, baby, **stay**	When I'm away from you, I miss your touch (ooh-ooh) You're the reason I **have** in love It's been difficult for me to trust (ooh-ooh) And I'm afraid that I'ma fuck it up Ain't no way that I can leave you stranded 'Cause you ain't ever left me empty-handed And you know that I know that I can't live without you So, baby, **stop**

Note that a *straight synonym replacement*, although it is a promising idea for detecting plagiarism or similar songs, *it may alter the meaning of the lyrics* as in our current example where an adverb (*like*) was replaced by a verb (*like*). It could also generate sentences that make no sense, such as "*I **have** in love*". Therefore, a synonym replacement must consider semantic and grammar aspects, as we point out in our future work.

Anyhow, we applied our method considering the lyrics of the song modified as explained. We left the other 540 songs unmodified. We tested with the following parameters: shingle size = 3, number of hash functions = 1024, and band size = 2. This time, the song got two matches; however, in both cases the shingles in common were *only two*:

- With the song "*Celebrate*" by Boyz II Men; the shingles in common were "*i feel love*" and "*reason i have*".
- With the song "*Fame or the Money*" by Tai feat. Authentic; the shingles in common were "*i feel love*" and "*so baby stop*".

Next, when we increased the band size to 4, we got one match with the song "*Celebrate*" by Boyz II Men, but *only one* shingle in common, "*the reason i have*".

After analyzing the lyrics of these three songs in more detail, no evidence of plagiarism was found.

With regard to emotions, after applying the WTA we got:

- "Stay" by The Kid LAROI and Justin Bieber: Fear and Tentative.

- "Celebrate" by Boyz II Men: Joy and Tentative.
- "Fame or the Money" by Tai feat. Authentic: Tentative.

In Figure 12 we show the WTA results for these songs.

Next, we show the tentative sentences of each song, according to the WTA:

"Stay" by The Kid LAROI and Justin Bieber:
- I told you I'd change even when I knew I never could.
- I feel like you can't feel the way I feel.

"Celebrate" by Boyz II Men:
- Cause I feel love... things that you say.
- Driving me crazy is nothing like my baby.
- Feel your arms around me.
- Cause I feel loved...

"Fame or the Money" by Tai feat. Authentic:
- I don't care 'bout the fame or the money.
- Think that you're not perfect.
- Stop believing your own lies.
- The way I know that you feel the same.
- Just know I will love you when push comes to shove.
- Now life's rough because I just can't get enough.
- Let's not speculate.
- So now I just wonder how we could discover.
- Whatever you need just call, no pressure.
- I guess I'm insane, I swear I'm not playing.

Thus, these three songs are not good candidates to be considered similar.

Next, in Table XII, we compare our proposal with applications that compare documents.

V. RELATED WORKS

In Table XIII, we review related works.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a song retrieval system based on the similarity of songs according to their lyrics (using minhash and LSH), emotions, and attributes such as artist and musical genre. Although minhash and LSH find possibly similar pairs of documents (from huge collections of documents), the nature of the documents should be considered. Thus, the degree of similarity required to consider that two lyrics are similar is not necessarily the same as that required for two scientific papers. We showed that this is the case in the examples of the introduction where two lyrics with only one or two shingles in common, are considered similar enough to lead to lawsuits.

According to the above, the results of our experiments did not show plagiarism (considering the few shingles in common detected among the pairs of songs). This finding suggests, at least from a textual point of view, that lyricists are careful on this matter. Nonetheless, more experiments with more sets of songs are necessary to further support this conclusion. We also validated the accuracy of our method by introducing songs into the set, two pairs of covers (or "almost" covers), which were detected by it. We also introduced some artificial songs.

Since minhash along with LSH detect similar elements, in our case based on identical fragments of lyrics, this process does not detect, e.g., plagiarism based on semantic aspects. Thus, e.g., if the lyrics of a song include the fragment "*she is not pretty*" and another song includes the fragment "*the girl ain't beautiful*", our system will show, effectively, that there are not shingles in common. Thus, additional work is necessary to deal with these situations (i.e., analysis of semantic similarity), e.g., through ontologies [38], where synonyms, generalization, metonymy are considered, among other semantic relationships. A domain ontology that contains idiomatic expressions and acronyms commonly used in lyrics may also be helpful. This approach should be coupled with normalization to standardize the English language as explained in Subsection IIIA.

Another future work will be to tackle the following cases. In [39] two lyrics are mentioned, one includes the fragment "*players gonna play, play, play, play, play*" and the other includes "*haters gonna hate, hate, hate, hate, hate*". Note that these lyrics (at least in these two fragments) do not show similarity either semantically or textually (except for the contraction "*gonna*"). Nonetheless, these two fragments were considered similar enough to cause a lawsuit.

With regard to emotions, we also plan to enrich our proposal with an associative classification approach [40], e.g., similarly to [41] where it is used to classify emails and identify spam emails. We could apply this approach for finding groups of songs that belong to the same classes, e.g., classes based upon emotions or another attribute. A similar work is [42] that applies several classifiers for labeling songs regarding six emotions (the authors used the "Emotions from Mulan" [43]: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, angry-fearful). Another option is to apply Dynamic Deep Learning [44] for finding positive and negative sentiments.

Finally, we plan to develop a more comprehensive procedure, possibly including a combination of several of the previous characteristics (textual, semantic, and audio similarity) to provide an even more accurate measure for the similarity between songs. Such a procedure could be complemented with a graphic representation where the similarities and their nature are highlighted. In addition, users should be able to set priorities based on how they want to determine the similarity, e.g., give more importance to the semantic aspect than to the textual one.

ACKNOWLEDGMENT

The authors thank Dr. David Rogers, Emeritus Senior Fellow, Kingston University, for grammar and style reviewing.

Fig. 12. WTA results: a) "*Stay*" by The Kid LAROI and Justin Bieber, b) "*Celebrate*" by Boyz II Men, and c) "*Fame or the Money*" by Tai feat. Authentic.

TABLE XII
COMPARISON WITH OTHER APPLICATIONS THAT COMPARE DOCUMENTS.

Feature	**Our proposal**	**Online *Text Compare Tool (Diff)* [1], *Text Compare!* [2], and *Diffchecker***	**Turnitin**
Does it compare several documents?	Yes	Yes, but only two at a time.	Yes
Does it show the differences between documents?	No	Yes. It highlights the differences between documents (letters, words)	Yes. It also highlights the similarities with other documents
Does it filter documents by attributes (e.g., genre, artist, emotions)?	Yes	No	No
Does the user have to enter the document to be compared?	No. The user only has to select the filters he/she wants to apply and the system will obtain similar documents.	Yes	Yes. A document is required to compare it with other documents on the web. The system returns links that lead to the original document.
Do results vary depending on the order of sentences in the document?	No	Yes	No

[1] http://www.ddginc-usa.com/spanish/text-compare-tool.html

[2] https://text-compare.com

TABLE XIII
RELATED WORKS

Reference	System or method	Description	Advantage
[12]	A music retrieval system that uses self-organizing feature maps (SOFMs) [13] and document level word embeddings followed by a baseline system that uses fuzzy c-means (FCM) [14] clustering.	A music retrieval system for Hindi songs that retrieves similar lyrics using SOFMs, the system preprocesses the datasets using an unsupervised stemming algorithm to normalize the lyrics.	The similar lyrics retrieval system can be combined with a metadata-based recommender to give a better performance. It is also useful for recommending a song where little or no metadata (genre, mood) is available.
[15]	A QBH System (*Query by Humming*) based on LSH.	A retrieval method called note-based LSH (NLSH) is proposed which is combined with pitch-based LSH (PLSH) for screening candidate audio fragments.	The method has better performance compared to similar methods considered there.
[16]	A system that represents the musical content of short pieces of audio based on the chroma [17] intervalgram.	An intervalgram is a summary of the local pattern of musical intervals in a segment of music. It supports LSH.	High precision matching with low false positive rate.
[18]	An AF (Audio Following) application that uses an index based on LSH to follow the position of the musician during his/her performance.	The AFP (Audio Fingerprint) of the reference performance is obtained. Then, the AFP is indexed using LSH.	High precision matching between the obtained alignment (of the live performance) and the reference alignment (of the reference performance).
[19]	A retrieval system based on the content of audio tracks using LSH.	LSH is applied to obtain compact and accurate representations of audio tracks.	The method achieved the best balance among storage, computation cost, and recall compared to similar methods considered there.
[20]	An algorithm that combines the properties of music (audio clips, humming, tapping, among others) in a compact signature through supervised learning.	An incremental LSH algorithm that supports retrieval by audio, by genre, tone, among others.	The algorithm facilitates tasks of musical retrieval, such as organizing, navigating, and searching in a dataset.
[21]	An algorithm to detect remixes of pop songs.	An algorithm that divides songs into fragments (audio shingles) and finds the similarities among them using the Euclidean LSH [22].	The algorithm recognizes similarities between a song and its remix.
[23]	A QBH system with filters.	The method has four filters: 1. LSH for finding possible similar candidates, 2. linear scaling [24] for eliminating false positives, 3. linear alignment [23] for locating the limits of the candidate, and 4. recursive alignment [25] for calculating the similarity of the melody.	It is stated that speed retrieval is improved compared to similar methods considered there.
[26]	A combination of algorithms to improve information retrieval based on audio tracks.	QBH by combining LSH and dynamic time warping (DTW) [27] algorithm.	The combination of LSH and DTW achieved a better balance between speed and precision than when LSH, DTW, and hidden Markov models (HMM) [28] were used alone.
[29]	A retrieval system based on musical content using chord progressions (CP) [30].	A three-phase algorithm is proposed: 1. Through musical rules, the CPs of the audio tracks are identified. 2. A summary of the song is obtained from the CPs. 3. The summarized tracks are organized by LSH.	The method is said to improve the accuracy and scalability of content-based music information retrieval, compared to other methods considered there.
[31]	An ethnic lyrics fetcher tool.	It uses the Google API to search for lyrics based on a song title and an artist.	Automatic lyrics search.
[32]	An automatic lyrics classification system using text mining techniques.	A system for the Thai language based on emotions.	Classification of the lyrics according to their emotions and generation of playlists.
[33]	Genre and mood classification using lyrics features.	Lyrics analysis using the POS (Part-of-Speech) [34] feature.	Classification of songs by genre and mood from the lyrics.

[35]	Lyric emotion estimation using word embedding learned from lyric corpus.	An emotion estimation method is proposed that detects lyric expressions that are not registered in emotions dictionaries.	The method does not depend on emotions dictionaries or labeled data.
[36]	Natural language processing (NLP) [37] of lyrics.	NLP algorithms like structure detection or text categorization.	Useful for training and validating algorithms based on audio.

APPENDIX

TABLE AI
LYRICS OF ONE OF THE MATCHES OBTAINED BY DISPENSING WITH THE GENRE AND EMOTIONS FILTERS

"*Thank You*" as recorded by Tori Amos	**"*Thank You*" as recorded by Led Zeppelin**
If the sun refused to shine *I would still be loving you* *If mountains crumble to the sea* *There would still be you and me* *Kind woman, I give you my all* *Kind woman, nothing more* *Little drops of rain whisper on the plains* *Tears have run thrust in the days gone by* *If our love is strong, here there is no wrong* *Together we shall go 'til we die* *Oh my, my* *Inspiration is what you are to me* *Inspiration, love to see* *If the sun refused to shine* *I would still be loving you* *If mountains crumble to the sea* *There would still be you, you and me* *And me*	*If the sun refused to shine* *I would still be loving you* *When mountains crumble to the sea* *There will still be you and me* *Kind woman, I give you my all* *Kind woman, nothing more* *Little drops of rain whisper of the pain* *Tears of loves lost in the days gone by* *My love is strong* *With you there is no wrong* *Together we shall go until we die, my, my, my* *An inspiration is what you are to me* *Inspiration - look, see* *And so today, my world, it smiles* *Your hand in mine, we walk the miles* *And thanks to you it will be done* *For you to me are the only one* *Alright, yeah* *Happiness, no more be sad* *Happiness - I'm glad* *If the sun refused to shine* *I would still be loving you* *When mountains crumble to the sea* *There will still be you and me*

TABLE AII
SHINGLES IN COMMON OF THE MATCH OF TABLE AI

Shingles		
"if the sun"	*"the sun refused"*	*"sun refused to"*
"refused to shine"	*"to shine i"*	*"shine i would"*
"i would still"	*"would still be"*	*"still be loving"*
"be loving you"	*"mountains crumble to"*	*"crumble to the"*
"to the sea"	*"the sea there"*	*"still be you"*
"be you and"	*"you and me"*	*"and me kind"*
"me kind woman"	*"kind woman i"*	*"woman i give"*
"i give you"	*"give you my"*	*"you my all"*
"my all kind"	*"all kind woman"*	*"kind woman nothing"*
"woman nothing more"	*"nothing more little"*	*"more little drops"*
"little drops of"	*"drops of rain"*	*"of rain whisper"*
"in the days"	*"the days gone"*	*"days gone by"*
"love is strong"	*"there is no"*	*"is no wrongv*
"no wrong together"	*"wrong together we"*	*"together we shall"*
"we shall go"	*"inspiration is what"*	*"is what you"*
"what you are"	*"you are to"*	*"are to me"*
"to me inspiration"		

TABLE AIII
ARTIFICIAL SONG

Fragment	Song source
Love, let me know, let me know *Love, let me know, let me know, let me know*	"*(At Your Best) You Are Love*" by The Isley Brothers
When I feel what I feel *Sometimes it's hard for me to tell you so* *You may not be in the mood* *To learn what you think you know*	
I mistook the warnings for wisdom *From so called friends quick to advise* *Though your touch was telling me otherwise* *Somehow I saw you as a weakness* *I thought I had to be strong* *Oh but I was just young, I was scared, I was wrong*	"*A Home*" by Dixie Chicks
Yeah, request: *I want everybody as close to the stage as possible!* *Get that speed, We're going back to the Heavyweight Jam* *Let's go out for a walk to the other side* *Get the sound, join the crew and you feel alright* *No more fiction go back to reality* *It's the message so listen and you will see* *No illusion the spirit is what you feel* *Get the volume tonite, you can make it real* *I explain once again, we won't let you down* *We can't stop going on that's what I pronounce* *Faster.....Harder.....Scooter!!!!* *We're getting Faster.....Harder.....Scooter!!!!* *We're getting Faster.....Harder.....Scooter!!!!* *We're getting Faster.....Harder.....Scooter!!!!*	"*Faster Harder Scooter*" by Scooter
I like to dominate *I create your fate* *Many years gone by* *I rule society* *You cannot be me* *I am the master*	"*All I Could Bleed*" by Testament
Can't you hear me, breathing for you *Do not ignore* *Reach out to me, put your knife through me* *Watch me bleed for you... yeah... right*	
Yah, anyway, wo na na *Weed* *High grade* *weed* *Good fi nerves* *weed* *Yeh* *Weed* *Hundred dollar bag* *weed* *Tell dem all about* *weed* *allright* *weed* *just me argument*	"*100 Dollar Bag*" by Beenie Man
Now I've had the time of my life *No I never felt like this before* *Yes I swear it's the truth* *And I owe it all to you* *'Cause I've had the time of my life* *And I owe it all to you*	"*(I've Had) the Time of My Life*" by Bill Medley and Jennifer Warnes
You gone take them 5 Or you gone take them to the trial *And go get denied* *By that probation and you just got, caught with that fry* *That alibi ain't gone work,* *Ain't it somethin' how them niggas from out that three be doin' that dirt* *Score a quarter from oh-oh, rock it up by four-four,* *Then you can go in them hallways and smoke that fire all day* *Shhhh....be quiet,* *Tonight is the night that we ride,*	"*3rd Ward Solja*" by Juvenile

Thirty camoflauge hummers with niggas inside *With choppers, doin', surgery on bodies like head doctors* *Be quiet, cuz they mad tonight, we gone act a ass tonight,* *I'ma take a body to that project for a sacrifice,* *That Calliope got that dope for less,* *Fuck around that bitch if you want, and get left,* *Brains hangin' off the steps, people cryin',* *Second line, T-shirtin', feet hurtin' from all of that twerkin'*	
Looking out on the morning rain *I used to feel so uninspired* *But when I knew I had to face another long, long day* *girl I used to feel so tired*	"*(You Make Me Feel Like) a Natural Man*" by Rod Stewart
Before the day I met you *life had been so unkind* *but you're the key to my peace of mind* *'Cause you made me feel, you made me feel* *you made me feel like a natural man*	

TABLE AIV
JOYFUL ARTIFICIAL SONG

Fragment	Song source
You know some days I like me. *Some days I don't.* *Some days I try with passion.* *Sometimes I won't.* *I might just hold my guard up,* *And lock my heart up tight,* *But it's the door that's open,* *Letting in the light.* *There's a battle raging* *Inside of me.* *It's a holy struggle,* *And it won't let go of me*	"*Fight*" by Amy Grant
Imy bad was not to let u in *when u stood by me better then a best friend* *and i thank u for givin me a hand* *now im ur biggest fan* *hercules superman* *ur #1 with a bullet baby* *when it comes to sparks ur making all the marks* *ur #1 with bullet honey* *in this heart of hearts* *uve been tearin up the charts*	"*#1 With a Bullet*" by Lindsay Pagano
Now I feel the time is right *Love will flow like wine tonight* *Give your love and it will come to you* *If you feel that you and me* *Could escape and hold the key* *To a paradise that's true and free* *Steal away, steal away* *Steal away - the night*	"*Steal Away (The Night)*" by Ozzy Osbourne
I like to dominate. I create your fate *Many years gone by. I rule society* *You cannot be me. I am the master* *Can't you hear me, breathing for you* *Do not ignore* *Reach out to me, put your knife through me* *Watch me bleed for you... yeah... right*	"*All I Could Bleed*" By Testament
All the love I miss loving, all the kiss I miss kissing. *All the love I miss loving, all the kiss I miss kissing.* *Before I met you baby, never knew what I was missing.* *All your love, pretty baby, that I got in store for you.* *All your love, pretty baby, that I got in store for you.* *I love you pretty baby, well I say you love me too*	"*All Your Love*" by Eric Clapton

REFERENCES

[1] I. Stav, "Musical plagiarism: a true challenge for the copyright law," *DePaul J. of Art Technology & Intellectual Property Law*, vol. 25, no. 1, pp. 1–52, 2014.

[2] A. Modupeoluwa, (2020, Jan 17). Ariana Grande sued for "7 Rings" lyrics plagiarism (Online). Available: https://guardian.ng/life/ariana-grande-sued-for-7-rings-lyrics-plagiarism

[3] R. Mowatt, (2019, Sep 24). Lizzo has been accused of plagiarizing "Truth Hurts" lyrics again (Online). Available: https://www.okayplayer.com/music/lizzo-truth-hurts-lyrics-plagiarism-details.html

[4] IBM, (2020, Sep 17). Watson Tone Analyzer (Online). Available: https://www.ibm.com/watson/services/tone-analyzer

[5] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*, Cambridge, UK: Cambridge University Press, 2020.

[6] The Music Lyrics Database, (2020, Oct 17). Club Tropicana lyrics (Online). Available: http://www.mldb.org/song-17813-club-tropicana.html

[7] The Music Lyrics Database, (2020, Oct 17). Loverboy Remix lyrics (Online). Available: http://www.mldb.org/song-5203-loverboy-remix.html

[8] M. A. T. Setiawan, "An analysis of the english grammar deviations in the 2016 song lyrics by african-american singers," Bachelor of Education thesis, Sanata Dharma University, Yogyakarta, 2018.

[9] The Music Lyrics Database, (2020, Oct 17). MLDb (Online). Available: http://www.mldb.org

[10] Oracle, (2020, Sep 17). Database application developer's guide - object-relational features (Online). Available: https://docs.oracle.com/cd/B19306_01/appdev.102/b14260/adobjcol.htm

[11] Allmusic, (2020, Sep 17). Music genres (Online). Available: https://www.allmusic.com/genres

[12] B. G. Patra, D. Das, and S. Bandyopadhyay, "Retrieving similar lyrics for music recommendation system," in *Proc. 14th International Conference on Natural Language Processing*, Kolkata, 2017, pp. 290–297.

[13] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 586–600, 2000.

[14] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: the fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2–3, pp. 191–203, 1984.

[15] Z. Guo, Q. Wang, G. Liu, and J. Guo, "A query by humming system based on locality sensitive hashing indexes," *Signal Processing*, vol. 93, no. 8, pp. 2229–2243, 2013.

[16] T. C. Walters, D. A. Ross, and R. F. Lyon, "The intervalgram: an audio feature for large-scale cover-song recognition," in *Proc. 9th International Symposium on Computer Music Modeling and Retrieval*, London, 2012, pp. 197–213.

[17] M. Kattel, A. Nepal, A. K. Shah, and D. Shrestha, "Chroma feature extraction," Department of Computer Science and Engineering, School of Engineering Kathmandu University, Nepal, 2019.

[18] L. F. Guzmán and A. Camarena-Ibarrola, "On the use of locality sensitive hashing for audio following," in *Proc. 19th Iberoamerican Congress on Pattern Recognition*, Jalisco, 2014, pp. 175–182.

[19] Y. Yu, M. Crucianu, V. Oria, and E. Damiani, "Combining multi-probe histogram and order-statistics based LSH for scalable audio content retrieval," in *Proc. 18th ACM International Conference on Multimedia*, Florence, 2010, pp. 381–390.

[20] B. Zhang, J. Shen, Q. Xiang, and Y. Wang, "CompositeMap: a novel framework for music similarity measure," in *Proc. of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Boston, 2009, pp. 403–410.

[21] M. Casey and M. Slaney, "Fast recognition of remixed music audio," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, 2007, vol. 4, pp. 1425–1428.

[22] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annual Symposium on Computational Geometry*, Brooklyn, 2004, pp. 253–262.

[23] Z. Guo, Q. Wang, L. Yin, G. Liu, and J. Guo, "Query by humming via hierarchical filters," in *Proc. 21st International Conference on Pattern Recognition*, Tsukuba, 2012, pp. 3021–3024.

[24] J. R. Jang, H. Lee, and M. Kao, "Content-based music retrieval using linear scaling and branch-and-bound tree search," in *Proc. IEEE International Conference on Multimedia and Expo*, Tokyo, 2001, pp. 1–4.

[25] X. Wu, M. Li, J. Liu, J. Yang, and Y. Yan, "A top-down approach to melody match in pitch contour for query by humming," in *Proc. 5th International Symposium on Chinese Spoken Language Processing*, Singapore, 2006, pp. 1–12.

[26] S. Zhou, Z. Zhao, P. Shi, and M. Han, "Research on matching method in humming retrieval," in *Proc. IEEE 3rd Information Technology and Mechatronics Engineering Conference*, Chongqing, 2017, pp. 516–520.

[27] Y. Kim and C. H. Park, "Query by humming by using scaled dynamic time warping," in *Proc. International Conference on Signal-Image Technology & Internet-Based Systems*, Kyoto, 2013, pp. 1–5.

[28] M. I. M. Yusoff, I. Mohamed, and M. R. A. Bakar, "Hidden Markov models: an insight," in *Proc. 6th International Conference on Information Technology and Multimedia*, Yogyakarta, 2014, pp. 259–264.

[29] Y. Yu, R. Zimmermann, Y. Wang, and V. Oria, "Scalable content-based music retrieval using chord progression histogram and tree-structure LSH," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 1969–1981, 2013.

[30] English Wikipedia (2021, Aug 27). Chord progression (Online). Available: https://en.wikipedia.org/wiki/Chord_progression

[31] R. P. Ribeiro, M. A. P. Almeida, and C. N. Silla Jr, "The ethnic lyrics fetcher tool," *EURASIP J. on Audio, Speech, and Music Processing*, vol. 2014, no. 1, pp. 1–10, 2014.

[32] C. Jareanpon, W. Kiatjindarat, T. Polhome, and K. Khongkraphan, "Automatic lyrics classification system using text mining technique," in *Proc. International Workshop on Advanced Image Technology*, Chiang Mai, 2018, pp. 1–4.

[33] T. C. Ying, S. Doraisamy, and L. N. Abdullah, "Genre and mood classification using lyric features," in *Proc. International Conference on Information Retrieval & Knowledge Management*, Kuala Lumpur, 2012, pp. 260–263.

[34] E. Brill, "A simple rule-based part of speech tagger," in *Proc. 3rd Conference on Applied Natural Language Processing*, Trento, 1992, pp. 152–155.

[35] K. Matsumoto and M. Sasayama, "Lyric emotion estimation using word embedding learned from lyric corpus," in *Proc. IEEE 4th International Conference on Computer and Communications*, Chengdu, 2018, pp. 2295–2301.

[36] J. P. G. Mahedero, Á. Martínez, P. Cano, M. Koppenberger, and F. Gouyon, "Natural language processing of lyrics," in *Proc. 13th Annual ACM International Conference on Multimedia*, Singapore, 2005, pp. 475–478.

[37] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. Cambridge, UK: MIT press, 1999.

[38] I. T. Afolabi, O. Y. Sowunmi, and T. Adigun. "Semantic text mining using domain ontology." In *Proc. World Congress on Engineering and Computer Science*, San Francisco, 2009, pp. 1-6.

[39] A. White, (2019, Oct 19). Taylor Swift's 'Shake It Off' copyright case headed back to court (Online). Available: https://www.nme.com/news/music/taylor-swifts-shake-off-copyright-case-headed-back-court-2562120

[40] F. Thabtah, "A review of associative classification mining," *The Knowledge Engineering Review*, vol. 22, no. 1, pp. 37–65, 2007.

[41] Rami Mustafa Mohammad, "An improved multi-class classification algorithm based on association classification approach and its application to spam emails," IAENG International Journal of Computer Science, vol. 47, no.2, pp187-198, 2020

[42] L. M. Gómez and M. N. Cáceres, "Applying data mining for sentiment analysis in music," In *Proc. International Conference on Practical Applications of Agents and Multi-Agent Systems*, Porto, 2017, pp. 198-205.

[43] K. Trohidis, G., Kalliris, G. Tsoumakas, G., and I. Vlahavas, "Multi-label classification of music into emotions,", *EURASIP Journal on Audio Speech and Music Processing* , vol. 2011, no. 1, pp. 325–330.

[44] Putra Wanda, and Huang Jin Jie, "DeepSentiment: finding malicious sentiment in online social network based on dynamic deep learning," IAENG International Journal of Computer Science, vol. 46, no. 4, pp616-627, 2019