UNIVERSITÀ
di VERONA

Computer Science Department
Mining massive datasets course: Detection of Music Plagiarism Project

Authors: Alessandro Lovo, Alessandro Pedron

GitHub - Repository

All the code present in this reposotory is made by us, since the article that we choose, even if the repository is linked, is no more acessible.

Abstract – **This is our final project for the course Mining massive datasets, based on the article Fuzzy vectorial based similarity detection of music plagiarism.**

This article provide a new approach in detection of music plagiarism. That consists in (1) its transformation in a vectorial representation, (2) retrieving of a list of similar melodies, (3) analysis and comparison with this subset of associated similar scores by using a fuzzy degree of similarity, that varies in a range between 0 for melodies that are fully musically different, and 1 for identical melodies.
Our goal is to understand and replicate the techniques used by applying them to different datasets and possibly make some improvements.

I. DATA - MIDI FILES STRUCTURE

The first thing that we focused on is finding the data, and understand the structure of the MIDI Files. Since in our article the GitHub repository is not present, we search the data in similar one focused in Detection of Music Plagiarism. So we use as reference this repository: GitHub - For data. They design four types of plagiarism for the dataset, taking into account the specific situations of plagiarism in reality: transposition, pitch shifts, duration variance, and melody change, where each accounts for a quarter of the dataset. The specific implications of these four situations are as follows:

- **Transposition**: structural transposition of original song.
- **Pitch Shifts**: pitch shifts of a fragment of the original song and the addition of it to a completely unrelated song.
- **Duration Variance**: a similar operation to pitch shifts, but with a duration variance to the original piece.
- **Melody Change**: melody change of a fragment of the original song, yet through MuseMorphose, a Transformer-based VAE model, and the addition of this fragment to a completely unrelated song.

This is not suitable with what is made in our paper, for this reason we take as dataset for our research only a part of the data contained in the repository that is called **datasetreal.py**, this dataset have 29 original songs and their equivalent plagiarism, contained in the dataset **datasetplag.py**. This are 29 'real-life' plagiarism songs, in this way we know that the two songs that we compare are for sure similar, and use the approach explained in the paper we can understand which part of the songs results plagiarized and in and to what extent.

A. STRUCTURE OF MIDI FILES

MIDI is a simple binary protocol for communicating with synthesizers and other electronic music equipment.
Opened the MIDI FILE, we find: **TRAK 0** that is the header of the song, that have the general information about it. Each song is composed by a series of messages contained:

```
note_on, channel=1, note=63,
velocity=90, time=18432
```

- **'Note on' - 'Note off'**: Tell us if a note is played or not played.

- **Channel**: These are used to turn notes on and off, to change patches, and change controllers (pitch bend, modulation wheel, pedal and many others). There are 16 channels, and the channel number is encoded in the lower 4 bits of the status byte. Each synth can choose which channel (or channels) it responds to. This can typically be configured.

- **Note**: Give us the conversion of the note in the pitch representation. In particular, each music note is associated to a pitch, a name and and octave position. Octaves are usually numbered from 0 to 6. In each octave there are 12 notes, named C, C# (or Db), D, D# (or Eb), E, F, F# (or Gb), G, G# (or Ab), A, A# (or Bb) and B.
  This is a table that represent this conversion:

Note	Octave 0	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5	Octave 6	Octave 7	Octave 8
C	12	24	36	48	60	72	84	96	108
C#	13	25	37	49	61	73	85	97	109
D	14	26	38	50	62	74	86	98	110
D#	15	27	39	51	63	75	87	99	111
E	16	28	40	52	64	76	88	100	112
F	17	29	41	53	65	77	89	101	113
F#	18	30	42	54	66	78	90	102	114
G	19	31	43	55	67	79	91	103	115
G#	20	32	44	56	68	80	92	104	116
A	21	33	45	57	69	81	93	105	117
A#	22	34	46	58	70	82	94	106	118
B	23	35	47	59	71	83	95	107	119

Fig. 1. Note - pitch conversion

- **Velocity**: Refers to the speed or force with which a note is played. In practical terms, velocity affects the loudness and sometimes the timbre of the note.
  - The velocity value ranges from 0 to 127.
  - A higher velocity value corresponds to a stronger, louder, or more pronounced note.
  - A lower velocity value corresponds to a softer, quieter, or less pronounced note.
- **Time**: A delta time is how long to wait before the next message. This time is express in ticks, "ticks" are time units used to measure the duration of musical events within a track. A tick represents a fraction of a standard musical note (such as a quarter note or a measure) and allows for precise and detailed representation of musical timing.

All this information is taken from Mido Library, where is possible to find also insights about it.

II. VECTORIAL REPRESENTATION

The first thing that we have to do is to convert the MIDI information about the song in a vectorial representation. Following the article we need to find:

- **VECTOR OF MELODY - M**: that is the conversion of note in the numeric representation, the pitch. In particular, each music note is associated to a pitch, a name and and octave position. Octaves are usually numbered from 0 to 6. In each octave there are 12 notes, named C, C# (or Db), D, D# (or Eb), E, F, F# (or Gb), G, G# (or Ab), A, A# (or Bb) and B.
  Example of M=(48,50,48,50,52,53,52,53).
  Since this information is already available in the messages that describe the songs **(Note)**, we simply extract them from this messages.
- **DELTA VECTOR - V_m**: That is the delta between the value of the vector M.
  Example of V_m = [2,2,+2,+2,+1,1,+1]
  Captures the sequence of interval values between consecutive notes in a melody. An interval is defined as the difference in pitch between two consecutive notes. Positive intervals indicate ascending motion, while negative intervals indicate descending motion.
- **NOTE DURATION - R_m**: This vector captures the duration of each interval represented in V_m. It contains the duration of each note in the melody, maintaining the same order as in V_m(how long is a 'note on'). We also better explain it through the code that we use to find it.
- **TUPLA DELTA, DURATION - (V_m, R_m)**: represents for each song the delta of the melody and the associated duration, we have to define it for each song that we want to compare, so we obtain: Song1: (V_{M1}, R_{M1}) and Song2: (V_{M2}, R_{M2})
- **SET OF SUBVECTORS OF $(V_{M1}$ and $V_{M2})$ - $S_k(V_{M1})$ and $S_k(V_{M2})$**: We don't want to compare for each song all the note, but we define a subvector $S_k(V_{M1})$ (Song1) and $S_k(V_{M2})$ (Song 2) of length k that we want to compare.
- **LENGTH OF THE SUBVECTOR - K**: Let denote $N_{M1,M2}$ the minimum number of notes contained in a measure of M_1 and M_2, and $M_{M1,M2}$ = min(V_{M1}size(),V_{M2}size()), and we set:

$$N_{M1,M2} < \mathrm{k} < M_{M1,M2}.$$

A. Why we have to find k?

We don't have general rules for identify cases of plagiarism but, based on our paper where they analyze some article, seems that we can divided the plagiarism in two categories:

- (1) text-like plagiarism class, which is the set of cases in which melodies have many similar parts (can be detected with text plagiarism detection approaches).
- (2) text-nolike plagiarism class, which is the set of cases in which melodies have very few similar parts (cannot be detected with text plagiarism detection approaches).

That can be easily see trough this example:

As an example of text-like plagiarism, we can consider "Love is a wonderful thing" of Micheal Bolton against "Love is a wonderful thing" of Isley Brothers. As we can see, in this case the melodies are very similar (also in the title!), and the court case ended with a sentence of plagiarism against Bolton.

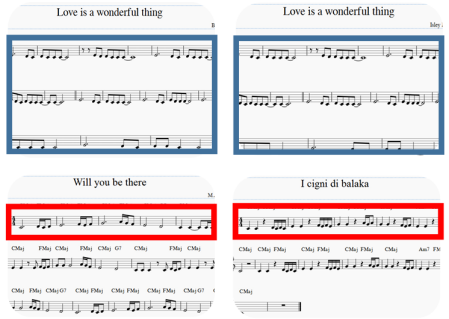


Fig. 2. Comparing text-like plagiarism and text-nolike plagiarism

As an example of text-nolike plagiarism, we can consider "Will you be there" of Micheal Jackson against "I cigni di Balaka" of two famous Italian singers: Albano Carrisi & Romina Power. In this case the melodies are very similar only

in the first three-four measures, but also in this case the court case ended with a sentence of plagiarism against Jackson.

Because of this based on what the article says we take only part of the vector melody and rhythm, based on the rule below, in order to solve this problem.

III. DISTANCES

Now we have all the element that we need in order to compare two songs, for do it we have to compute the **Overall distance** D(s_1, s_2) between two melodic sub-sequence s_1 and s_2, that are extracted from the melodies represented vectorially in V_{M1} and V_{M2}. The Overall distance is composed of two main components:

- **VECTORIAL MELODIC DISTANCE -** $D_m(s_1, s_2)$:

$$\sqrt{\sum_{i=1}^{k} |ni \cdot V_{M1}.getPos(ni) - mi \cdot V_{M2}.getPos(mi)|^2}$$

  This metric gives us a measure of the difference between the two melodic sequences in terms of their pitches and the positions of those pitches in the sequences. A smaller value of $D_m(s_1, s_2)$ indicates that the sequences are more similar in their melodic content.

- **VECTORIAL RHYTHMIC DISTANCE -** $D_r(s_1, s_2)$

$$\sqrt{\sum_{i=1}^{k} |di \cdot R_{M1}.getPos(di) - ti \cdot R_{M2}.getPos(ti)|^2}$$

  This metric provides a measure of the difference between the rhythmic patterns of the two sequences. A smaller value of $D_r(s_1, s_2)$ indicates that the sequences have more similar rhythmic structures.

Compute this two element $D_m(s_1, s_2)$ and $D_r(s_1, s_2)$ is necessary in order to obtain $D(s_1, s_2)$=($D_m(s_1, s_2)$, $D_r(s_1, s_2)$) that combining it together give us a comprehensive view similar to considering both spatial and temporal dimensions when comparing objects or events. Since $D_m(s_1, s_2)$ take in count both pitch and rhythm offering a complete understanding of the relationship between the sequences.

When comparing multiple pairs of sequences, the pair with the smallest D_m and D_r values is considered the most similar. If there are ties in D_m, the pair with the smallest D_r value is chosen.

A. Note

This two distance is essentially calculating the **Euclidean distance** between two vectors in a multi-dimensional space where each dimension corresponds to the pitch value weighted by its position in the sequence. (Same of $D_r(s_1, s_2)$ but the dimension is the rhythmic value (duration) weighted by its position in the sequence.)

IV. SIMILARITY - JACCARD COEFICCIENT

The Jaccard similarity is a statistical measure used to compare the similarity/diversity of sample set. In our context, music plagiarism detection, the Jaccard similarity can be utilized to compare two pieces of music to determine how similar they are, which helps in identifying potential plagiarism.

The general formula is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard index assumes values that vary from

$$0 < J(A,B) < 1$$

where if $J(A,B) = 0$ Indicates that there are no common elements between the two pieces, instead if $J(A,B) = 1$ indicates that the two pieces share all the elements, suggesting total similarity.

But in our contex, we have to to manage all the element that we found until now, that is:

- **Melodic subset -** $S_k(V_{MA})$**,** $S_k(V_{MB})$**:** That represent the subset of length k of the vectors V_{MA} and V_{MB}
- **Rhythm subset -** $S_k(R_{MA})$**,** $S_k(R_{MB})$**:** That represent the subset of length k of the vectors R_{MA} and R_{MB}

Using this element we compute:

- **Melodic Jaccard similariry:**

$$J(S_k(V_{MA}, S_k(V_{MB}))_m = \frac{|S_k(V_{MA}) \cap S_k(V_{MB})|}{|S_k(V_{MA}) \cup S_k(V_{MB})|}$$

  That provides a quantitative measure of the similarity between two musical pieces based on their melodic elements.

- **Rhythm Jaccard similariry:**

$$J(S_k(R_{MA}, S_k(R_{MB}))_r = \frac{|S_k(R_{MA}) \cap S_k(R_{MB})|}{|S_k(R_{MA}) \cup S_k(R_{MB})|}$$

  That provides a quantitative measure of the similarity between two musical pieces based on their rhythm elements.

Combining this two results we are able to detect which part of the song seams to be plagiarized, and if this plagiarism is due to a melodic similarity or to a rhythm similarity or both.

V. AVERAGE JACCARD SIMILARITY

Why we compute the **average jaccard similarity** ?

Calculating the average melodic Jaccard similarity coefficient between M_x and M_y serves to obtain a more precise and detailed measure of melodic similarity between the two melodies.

But first of all we need to define M_x and M_y:

M_x **is suspicious melody**, refers to a melody under investigation for potential plagiarism. This is the melody for which we want to find similar melodies in a collection of melodies C_M. The process involves comparing M_x with each melody $M_y \in C_M$ to determine similarities.

A. CLARIFICATION - *DIFFERENCES BETWEEN JACCARD SIMILARITY and AVERAGE JACCARD SIMILARITY*

- **JACCARD SIMILARITY** Measure the similarity between two sets by considering the ratio between the intersection and the union of the sets. In our context, we calculated between the $S_k(V_{M1})$ and $S_k(V_{M2})$ (subsequences of k notes) of the vectorial representations of the melodies. **It is calculated once considering the entire melody (and entire rhythm), providing an overall view of the similarity.**
- **AVERAGE JACCARD SIMILARITY** This measure considers different sections of the melody, calculating the Jaccard similarity for different k-shingle and then averaging these values. **This approach captures local variations in similarity that may not be evident when considering only an overall measure. In other words, the average melodic Jaccard similarity can detect similarities in specific parts of the melodies that might be missed in a global evaluation.**

  - 1) Melodic - $J(Mx,My)_m$:

$$\frac{\sum_{N_{Mx,My} \leq k \leq M_{Mx,My}} J(S_k(V_{Mx}), S_k(V_{My}))_m}{M_{Mx,My} - N_{Mx,My} + 1}$$

  - 2) Rythm - $J(Mx,My)_r$:

$$\frac{\sum_{N_{Mx,My} \leq k \leq M_{Mx,My}} J(S_k(R_{Mx}), S_k(R_{My}))_r}{M_{Mx,My} - N_{Mx,My} + 1}$$

B. SUMMARY OF THE PROCEDURE

- Divide the melody in k shingle: The melody is divided into different sections or k-shingle. For example, a melody of length n, could be divided into k-shingle of length k.
- Calculation of Local Jaccard Similarities: The Jaccard similarity is calculated for each pair of k-shingle of the two melodies M_x and M_y.
- Average of similarity: The local Jaccard similarities are then averaged to obtain the average melodic Jaccard similarity coefficient.

Average Jaccard coefficient, check tuple of shingles s_x with the same length k taking from $S_k(V_{Mx})$, compute the Jaccard similarity for all the tuples (s_{x1}, s_{y1}) until (s_{x1}, s_{yn}) fixing s_{x1} (shingle of the suspected song) and letting vary s_{yn} all the shingles of the other song, for different songs.

Example of the procedure: After the vectorial composition we will have $S_k(V_{Mx})$ and $S_k(V_{My})$ that is the subvectorial representation of V_{Mx} and V_{Mx}, for each subvectorial representation, we obtain n shingles of length k (where k is choosen by us, in this case, must be lesser that the length of $S_k(V_{Mx})$ and $S_k(V_{My})$). From that we obtain $shingles_x = [s_{x1}, \ldots, s_{xn}]$ and $shingles_y = [s_{y1}, \ldots, s_{yn}]$, so we pick all the tuple of shingles of the two songs that we have to analyze, $(s_{x1}$ and $s_{y1})$ that are respectively the first shingle of the suspected song and the firt shingle of a song we want to compare, until the last shingle $(s_{x1}$ and $s_{yn})$, we fix the s_{x1} shingle for the suspected song and we let vary all the shingles of s_y. For each tuple we compute Jaccard similarity, and we save the max value that we found in order to give weight to the maximum similarity found. After that we repeat the process all the other shingles of s_x. And as last step we sum all the result that we obtain and we divided by the number of comparisons that we made,**getting the Average Jaccard similarity (computed for both Melodic and Rhythm vectors)**.

VI. FUZZY VECTORIAL - BASED SIMILARITY APPROACH

This approach improves music plagiarism detection, by integrating the 'fuzzy logic', that consist in detailed vectorial representation, granular similarity measures. These enhancements ensure a more accurate, precise, and nuanced analysis of musical similarities (given by the computation of the average jaccard similatiry computed in the k-shingles), effectively identifying potential plagiarism while accommodating the complexities and subtleties of musical compositions.

A. *How does it works ?*

The first thing that we have to do is defining the List of similar songs L_{M_x}, in this list we put all the songs that have the Average jaccard similarity greater than a fixed threshold for the melody $J(M_x,M_y)_m > t_m$ and for the rhythm $J(M_x,M_y)_r > t_r$. Now we have all the songs that we consider similar in terms of melody and rhythm, but we are not satisfied with this. For each songs $M_y \in L_{M_x}$ (M_y is a melody similar to M_x which passed the threshold), we obtain the shingles of length k, as explained before, and what we do is fixing a shingle s_{x1} from the suspected song and we compare it with all the shingles $shingles_y = [s_{y1}, \ldots, s_{yn}]$ of another song. For every comparison we compute the normalize distance D(s_x,s_y), from all the tuple of shingles, this value vary from [0,1]. This is necessary because we have to compute the **fragment to melody correlation factor**:

$$\lambda(s_y, M_x) = 1 - \prod_{s_x \in S_k(V_{Mx})} (1 - d(sy, sx))$$

This formula tell us $\lambda(s_y, M_x)$ correlation factor between a musical fragment s_y from melody M_y and the entire melody M_x, where $\prod_{s_x \in S_k(V_{Mx})}(1 - d(sy, sx))$ represents the product of the complements of the normalized distances between s_y and each subvector s_x in $S_k(V_{M_x})$ This product measures how s_y collectively differs from all the subvectors of M_x.

If at least one of the s_x is very similar to s_y (low distance), the complement of this distance will be high.

Subtracting this product from 1, we obtain a value that reflects the overall similarity. If s_y is very similar to one or more s_x, the product will be close to zero, so 1 - product will be close to 1, indicating high similarity.

Conversely, if s_y is dissimilar to all s_x, the product will be close to 1, so 1 - product will be close to 0, indicating low similarity.

In conclusion we can say that $\lambda(s_y, M_x)$ quantifies the similarity of a single fragment s_y relative to the entire melody M_x based on the normalized distances between the fragment and all the subvectors of the melody. This approach captures local similarities within melodic structures, contributing to an overall measure of similarity between melodies.

Now finally we can compute the **fuzzy vectorial - based similarity** between M_x and M_y as a fuzzy number between 0 and 1 using the following equation:

$$F(M_y, M_x) = \frac{\sum_{s_y \in S_k(V_{M_y})} \lambda(s_y, M_x)}{V_{M_y}.size()}$$

Where the $\sum_{s_y \in S_k(V_{M_y})} \lambda(s_y, M_x)$ represents the aggregate of individual similarities between each fragment s_y of melody M_y and melody M_x.

Each $\lambda(s_y, M_x)$ measures how similar the fragment s_y is to the entire melody M_x. By summing these measures, we get an overall indication of the similarity between all parts of M_y and M_x.

Dividing by $V_{M_y}.size()$ normalizes this sum, producing an average of the fragment similarities. This ensures that the overall similarity is a value between 0 and 1, regardless of the length of the melodies.

$F(M_y, M_x)$ provides an overall measure of similarity between two melodies. It takes into account the similarity of each fragment of melody M_y to the entire melody M_x, combining these pieces of information into a normalized average. This approach allows for a granular and detailed assessment of similarity, considering both global and local characteristics of the compared melodies.

B. Note about FUZZY VECTORIAL - BASED SIMILARITY APPROACH

One of the main differences between this approach and the average Jaccard similarity is that the average Jaccard similarity is computed on both the subvector representation of the melody and the rhythm, and is based on the Jaccard coefficient. In contrast, the fuzzy vectorial similarity approach uses the Jaccard similarity only to determine if a song reaches a certain level of similarity, considering a fixed threshold to be included in the list L_{M_x}. After this, it takes into account only the subvector representation of the melody of the songs in the list L_{M_x} (ignoring the vectorial representation of the rhythm).

VII. EXPERIMENTAL ANALYSIS

In this section we test the **Fuzzy vectorial - based similarity approach** using as data the 29 songs that represent a 'real - life' cases of plagiarism. So our aim, using this data, is not to detect if a song M_x is consider a plagiarism of a song M_y, since a legal sentence already provides us this information. Our goal is to quantify how similar a song is to another in terms of melody and rhythm, after this we create a list of song L_{M_x} that are those songs that pass a threshold for the melody $J(M_x, M_y)_m > t_m$ and for the rhythm $J(M_x, M_y)_r > t_r$ and for those that are in the list, we apply another measure of similarity that is the **fragment to melody correlation factor**, that compose the **fuzzy vectorial - based similarity**.

A. RESULTS OF JACCARD SIMILARITY - AVERAGE JACCARD SIMILARITY AND FUZZY VECTORIAL - BASED SIMILARITY APPROACH

In this section we want to show some results given by the application of the Jaccard coefficient for the melody J_{V_m} and for the rhythm J_{V_r}, the Average Jaccard coefficient $Avg\mathbf{J}_{V_m}$ for the melody and $Avg\mathbf{J}_{V_r}$ for the rhythm computed from the 29 'real -life' case songs. And also for the song that pass the threshold for the melody $J(M_x, M_y)_m > t_m$ where t_m = 0.2 and for the rhythm $J(M_x, M_y)_r > t_r$ where t_r = 0.15 , we insert this in the list of similar songs L_{M_x}. We compute the **Fuzzy vectorial - based similarity** F_My_Mx for the 12 songs inside L_{M_x}, and this is our results (fixing the length of the shingles k = 5).

Case N.	J_Vm	J_Rm	Avg_J_Vm	Avg_J_Rm	F_My_Mx
10	0.455	0.500	0.806	0.956	0.950
11	0.750	0.000	0.969	0.000	NaN
12	0.357	0.000	0.762	0.000	NaN
13	1.000	0.000	1.000	0.000	NaN
14	0.333	0.000	0.250	0.000	NaN
15	1.000	1.000	1.000	1.000	0.543
16	0.158	0.429	0.146	0.311	NaN
17	0.533	0.700	0.835	0.678	0.800
18	0.800	0.000	0.680	0.000	NaN
19	0.556	0.600	0.575	0.379	0.822
1	0.786	0.714	0.711	0.871	0.925
20	0.714	0.400	0.917	0.823	NaN
21	0.565	0.467	0.507	0.398	0.905
22	0.176	0.278	0.300	0.314	0.789
23	1.000	0.000	0.975	0.000	NaN
24	0.778	0.000	0.934	0.000	NaN
25	0.310	0.467	0.300	0.409	0.766
26	0.727	0.000	0.731	0.000	NaN
27	0.727	0.000	0.904	0.000	NaN
28	0.583	0.588	0.814	0.628	0.820
29	0.237	0.500	0.175	0.383	NaN
2	0.625	0.667	0.648	0.963	0.606
3	0.529	0.000	0.815	0.000	NaN
4	0.850	0.733	0.804	0.804	0.928
5	0.812	0.000	0.793	0.000	NaN
6	0.706	0.278	0.743	0.819	0.440
7	0.429	0.007	0.456	0.087	NaN
8	0.765	0.000	0.859	0.000	NaN
9	0.727	0.000	0.775	0.000	NaN

After this we fix a threshold α = 0.75 for the **Fuzzy vectorial - based similarity coefficient**, and we obtain only 9 songs in this table:

Case N.	Ori_Song	Plag_Song	F_My_Mx
10	Anftrag Deutsches Reich-Stahlgewitter	Schenkt uns Dummheit, kein Niveau	0.950
17	Ma este meg	Perhaps	0.800
19	Phantom Song	Till you	0.822
1	Song_a	Song_b	0.925
21	Here We Are On the Road Again	Family Welcome	0.905
22	City of Violets	Chariots of Fire	0.789
25	There is nothin like a dame	Advert for Rapid Coach Service	0.766
28	He so fine	My sweet lord	0.820
4	十月里响起一声春雷	我爱你中国	0.928

B. Observation

From the first table we can notice that eaven if we have high value of the Average Jaccard similarity, the Fuzzy vectorial based coefficient is not correlated. For example we can notice from the **Case 15** where we have:

Case N.	Avg_J_Vm	Avg_J_Rm	F_My_Mx
15	1	1	0.543

The song seems to bee equal in terms of melody and rhythm, but the Fuzzy vectorial based coefficient give us as result $F(M_y, M_x)$=0.543.

And for the opposite situation we have **Case 19**:

Case N.	Avg_J_Vm	Avg_J_Rm	F_My_Mx
19	0.575	0.379	0.822

In reality this is not a real surprise, since as we explained above, this two coefficient depends on different factors, one is based on the computation of the jaccard similarity reiterating the process for all the shingles. The other is based on the computation of the fragment to melody correlation factor $\lambda(s_y, M_x)$, used for composed the fuzzy vectorial - based similarity coefficient.

For the **Case 20**, we have:

Case N.	Avg_J_Vm	Avg_J_Rm	F_My_Mx
20	0.917	0.823	NaN

Even if $J(M_x, M_y)_m > t_m$ where t_m = 0.2 and for the rhythm $J(M_x, M_y)_r > t_r$ where t_r = 0.15, we have that the F_{M_y,M_x}= NaN, because of the length of the shingles (k=5 in this case). The NaN value is due to the fact that the denominator of the fragment to melody correlation factor is equal to zero, since the normalized distance take as max value(0 in this case). For solve this problem one possible solution is to increase the length of the shingles.

VIII. OUR IMPROVEMENTS

In this section we want to give our contribution on the paper the we analyze. If until now we follow the path that is design in it (even if we implement it using a complete different dataset), now we want to try to go off the rails, in order to add our contribution.

The first idea that we try to develop is to apply the **Minhash** to the melody and rhythm vector. In order to verify if is possible to improve the result that we obtain with the Naive approach using the Minhash, and we want to compare also execution time with this these two techniques.

The second suggestion, check if the length of the shingles influence the result that we obtain (Jaccard similarity, average Jaccard similarity and Fuzzy vectorial based coefficient).

A. Minhash

Minhashing is a powerful technique for estimating the similarity between sets by reducing the complexity of comparison through compact, approximate representations. This makes it a valuable tool in various applications where handling large sets or datasets efficiently is crucial.

1) Procedure: The first thing that we have to do is to obtain the shingles s_m from the melody V_m, so to do it we have to set a shingles length (same for the rhythm vector R_m). Now we have to hash each shingles, and we obtain for each song an hash representation of it, at this point we are able to apply the Minhash. The hash function will take this form: $h(x) = (a * x + b)\%c$ where 'x' is the input value, 'a' and 'b' are random coefficients, and 'c' is a prime number greater than max shingle ID. Now we have to obtain the coefficient, that returns the first k element of a random permutation. Rather than generating a random permutation of all possible shingles, we'll just hash the IDs of the shingles that are 'actually in the document', then take the lowest resulting hash code value. This corresponds to the index of the first shingle that you would have encountered in the random order.

2) Jaccard with minhash: Now we can compute the Jaccard coefficient where we compare the shingles that we obtain after the application of the minhash.

3) Conclusion: We expected that the result will be similar to what we obtain with the Naive approach, but we see that it's not like that. One possible explanation, is the fact that the Minhash technique is particularly effective when we have large data, in this case the vector of the melody and the rhythm doesn't fit. For this reason the result that we obtained are very different from those that we obtain from the Naive approach. We also know that one of the strengths of the Minhash is the computational reduction time, so we computed for both technique and we obtain that the computational time is almost the same. We can conclude that the Minhash approach doesn't fit for this dataset and for the detection of similarity in music field.

Note: We also try to increase the number of permutation in order to improve the results of the Minhash, but this give us again poor results.

B. Different length of k

In this section we want to very if there is a linkage between the dimension of the shingles and the results of the Jaccard similarity, the average Jaccard similarity and the Fuzzy vectorial based coefficient.

What we do is to simulate the results, letting vary the length of k, starting from a shingle length of 5 until the max length of the of the subvector V_m (same V_r). What we appreciated from this simulation is that for the Jaccard and average Jaccard coefficient, the results doesn't show a tendency, it's seems that the length of the shingles influence randamly the results. Instead for the fuzzy vectorial based coefficient, appear that if we increase the length of the shingles, the value of the coefficient decrease. We have an inversely proportional tendency.

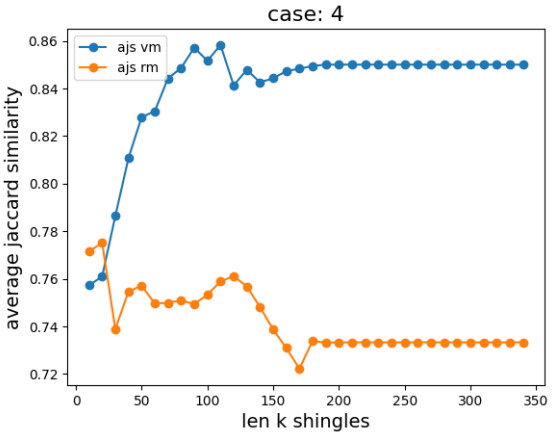
1) Influence of Shingle Length on Jaccard Similarity, Average Jaccard Similarity, and Fuzzy Vectorial Coefficient: The length of the shingles k affects the outcomes of the three similarity indices Jaccard Similarity, Average Jaccard Similarity, and Fuzzy Vectorial Coefficient in different ways. Below is an explanation of how varying shingle length can impact the results for each index:

- **Jaccard Similarity**, measures the ratio of the intersection to the union of two sets of shingles. The length of the shingles k can affect the intersection and union of these sets in a non-linear manner. For small values of k, there is a higher likelihood of overlap between the shingles of two similar strings, increasing the intersection and thereby the similarity. However, as k increases, the shingles become more specific, reducing the likelihood of random overlaps, which may decrease the intersection and thus lower the similarity. This could explain why no regular trend is observed: the behavior of Jaccard Similarity may vary depending on the structure of the compared strings.
- **Average Jaccard Similarity**, computes the average of the Jaccard Similarities between multiple pairs of segments within a string. Since this measure is an average, its behavior might be less volatile than the simple Jaccard Similarity. However, the shingle length k can still influence the result in the same way as described above: longer shingles tend to be more specific, which can reduce the average similarity if the compared strings do not have exactly matching segments. The lack of a regular trend here might be due to the variation in the individual Jaccard Similarities that are averaged.
- **Fuzzy Vectorial Coefficient**, measures similarity by considering a vectorial representation of the melodies. This measure is more sensitive to the length of the shingles. As k increases, the shingles become more specific, and the system might detect more pronounced differences between the vectors. This leads to a reduction in the fuzzy coefficient value, as the distance between the vectors increases when the representations are more detailed. This behavior explains the observed trend: as the shingle length increases, the Fuzzy Vectorial Coefficient tends to decrease.

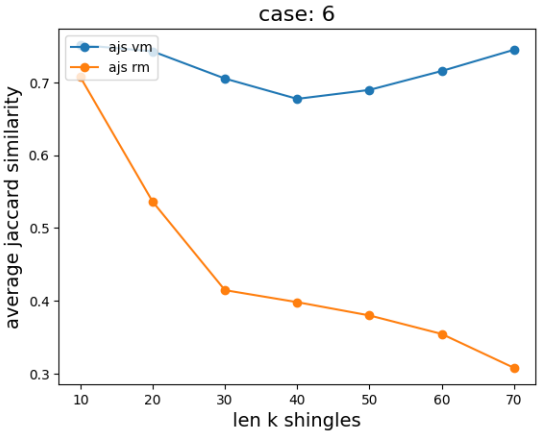
Below some graphs that show what has been said:

2) Average Jaccard similarity with different k:

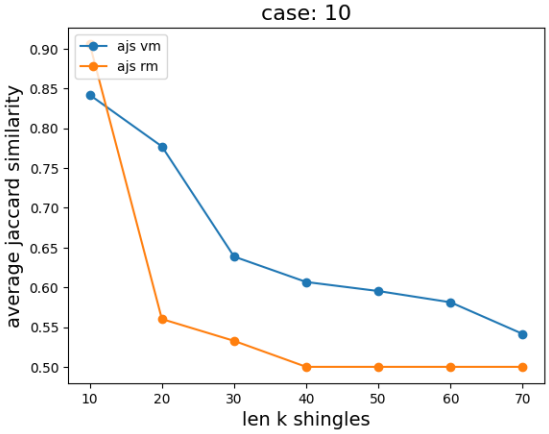
- Case 4:



- Case 6:

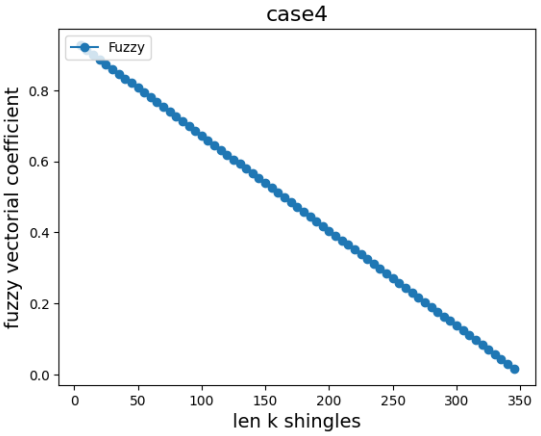


- Case 10:

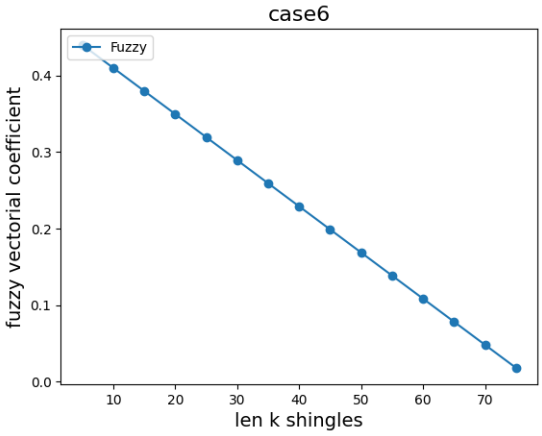


3) Fuzzy vectorial based coefficient with different k:

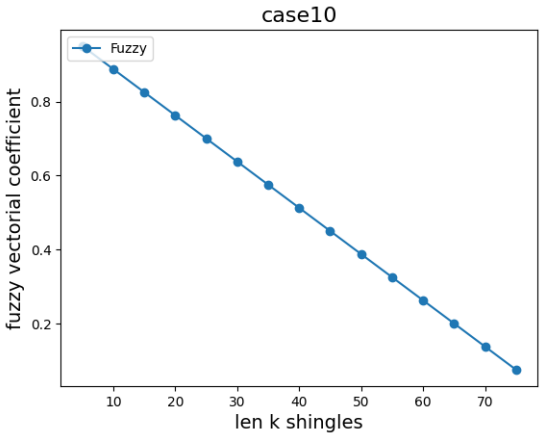
- Case 4:



- Case 6:



- Case 10:



IX. CONCLUSION

At the end we test our algorithm with the bigger dataset, the one the we describe in DATA - MIDI FILES STRUCTURE, with 250 songs. In particular we only take the MPD-Set in the melody folder, where we have the original song and **melody change of a fragment of the original song**. In this way we are able to compare the original song with his melody plagiarism, following the same procedures that we made for the 29 'real case' songs, in order to obtain in percentage of how many songs pass an α thresholds, that we let vary from 0.25 to 0.75 (Remark: this thresholds is applied on the result of the fuzzy vectorial based coefficient).
Taking inspiration from the article we fix different thresholds t_m, t_r for the melodic and the rhythm average jaccard similarity $AvgJ_m$ and $AvgJ_r$. And we also letting vary the threshold α concerning the Fuzzy vectorial based coefficient $F(M_y, M_x)$. This leads us to conclude, what is percentage of cases of plagiarism detected based on different thresholds.

t_m	t_r	α	%
0.2	0.15	0.75	64%
0.15	0.2	0.5	86%
0.25	0.15	0.35	98%
0.1	0.1	0.25	99%
0.2	0.2	0.75	62%

Our outcome, based on our data is that the average jaccard similarity, based on the melody and the rhythm, it's not a real obstacle but it helps us to delete those songs that don't seem to be similar at all, instead the real discriminant is the threshold α which changes the results significantly. The most evident case is the first and the last row, where there is the higher value of α=0.75 and the lowest percentage of songs that pass the threshold.

In this report we try to replicate the analysis of the paper Fuzzy vectorial based similarity detection of music plagiarism, where is underlined how there are gaps in the legislation regarding the matter of plagiarism. They try to propose a new algorithm that solves these problem, at least in part. We also add our contribution, trying to understand how the length of the shingles affected the results and if it was possible to apply the Minhash technique, that we saw during the course to verify the similarity between text documents. We see that this technique improved efficiency compared to the Naive approach, in computational terms and space used, but is not verify for our data. We specify that this method should be tested with different data to conclude its total inefficiency, we report what was tested with the available data.

REFERENCES

- Fuzzy vectorial based similarity detection of music plagiarism, by Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino and Rocco Zaccagnino
- GitHub Repository for MIDI data: GitHub - MIDI data
- "Mining of Massive Datasets", by Jure Leskovec, Anand Rajaraman, Jeff Ullman
- "Data Mining: The Textbook", by Charu C. Aggarwal
- Slide Mining Massive Datasets, by Professor Damiano Carra