Journal of Information Hiding and Multimedia Signal Processing ©2023 ISSN 2073-4212
Ubiquitous International Volume **14**, Number **2**, June **2023**

Detecting Music Plagiarism Based on Melodic Analysis

Phuc Nguyen[1,2], Hao Le[1,2], Van Bui[1,2], Thao Chau[1,2], Vy Tran[1,2], Tam Bui[1,2]

[1]Faculty of Information Systems, University of Economics and Law, Ho Chi Minh City, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam
phucnq@uel.edu.vn, {haolc, vanbt, thaocnh, vytt, tambtt}20416c@st.uel.edu.vn

Received April 2023; revised June 2023

Abstract. *Copyright protection is getting more and more attention, and with an industry that is so pervasive to the general public like the music industry, the problem of plagiarism is a controversial topic. However, detecting whether a work is infringing on copyright is still a difficult problem because of the limitation in the verification work. Besides that, music plagiarism is a broad category. So there are numerous ways for an individual or organization to cause plagiarism, including sampling, rhythmic, and melodic methods. At the same time, the level for determining how many similarities between two songs is considered plagiarism is also a major obstacle. In this work, we attempted to detect music plagiarism by melodic methods: Supervised learning algorithm, Edit distance, and N-grams. We also present a new dataset of real legally-judged music plagiarism cases and conduct detailed court studies to be more objective. In progress, we conduct audits to verify which methods are really effective. Finally, we recommend the best method for developing the project's graphical user interface. Future works will be an improvement of these methods, promising the usefulness of automatic tools that provides a measure of similarity score between songs.*
Keywords: Music plagiarism detection, Melodic analysis, Bipartite graph matching.

1. **Introduction.** Music plagiarism is known as the act of copying existing work which is protected by copyright without permission. Two things must be demonstrated in order to establish that your melody has been lifted. The first thing to prove is that the defendant had "access" to your music. The second point to demonstrate is the "substantial similarity" of the melodies [1]. Many famous singers were sued for suspicion of plagiarism such as Justin Bieber, Katy Perry, Ed Sheeran, etc. Perry was alleged in 2014 of plagiarizing Marcus Gray's song Joyful Noise for her hit Dark Horse, which was the second best-selling song in the world that year. Katy Perry eventually won an appeal in a copyright case involving 2.8 million dollars in damages. However, a brief coincidence happens. Not only can plagiarism be detected through rhyme, but it also can be detected through lyrics. The outstanding songwriter Taylor Swift was accused of getting ideas for lyrics for "Shake It Off". It can be challenging to define similarity properly, and music similarity is still among the most challenging issues in retrieving data in the field of music. This idea might be heavily influenced by musical culture, individual preferences, mood, etc. [2]. When allegations of music plagiarism are brought before a court, independent music professionals will assess the similarities between two songs and make a determination based on their personal opinions [3]. A technology that can automatically identify song similarities will help the music expert swiftly and accurately identify plagiarism. This paper presents automatic tools based on music theories and applies three approaches: String Matching

Algorithm: Edit distance and String Matching Algorithms combine Graph theory (Bipartite Graph Matching), N-gram algorithm, Supervised Learning Algorithm: K-Nearest Neighbour (KNN) and Support Vector Machine (SVM). We experiment with the above methods to find a suitable model for detecting plagiarism and then choose the most optimal model with the available data set. In the future, the research will expand the process of musical plagiarism detection by analyzing more features of melodies and extending the variation of input song genres to increase the accuracy of the algorithms.

2. **Literature.**

2.1. **Musical common knowledge.**

- **Melody** is created when multiple musical notes operate as a single unit. And identifying note sequences that are likely to match the perceived melody is the goal of a melody extraction approach [5].
- **Musical Instrument Digital Interface (MIDI)**: MIDI is a data communications protocol, a contract between makers of music systems, computers, and software that outlines how information and control signals can be sent between music systems and related equipment [6].
- **Music plagiarism**: A melody can be considered identical even if a musical idea is transposed to another key, slowed down, speed up or interpreted with different rhythmic accentuation. Plagiarism may be detected when successive melody notes in two pieces of music are identical [7]. Despite the fact that there are other approaches to detect plagiarism in music, including sampling, rhythmic, and melodic methods [3, 20], the majority of studies employ melodic as the key characteristic derived from their separate datasets because according to the study. Since it may result in a false impression about whether the music is plagiarized or not, rhythmic extraction alone won't be helpful in identifying plagiarism in music [19]. Therefore, it is preferable to extract the song's melody since it has the best structure for producing precise results.

2.2. **Melodic technique methods.** Finding musical commonalities has been tackled by several algorithms. The most popular and effective melodic techniques will be the main topic of this essay.

2.2.1. *Edit Distance.* When comparing two strings in computer science, the edit distance is determined by calculating the smallest number of operations required to change one string into the other. Various string operations are used depending on the edit distance. The deletion, insertion, or substitution of a character in the string constitutes the standard distance operations. Assume that we wish to determine the edit distance between two strings, x and y, each of which has a length of n and m. Initialize a matrix of dimension d, where n is the length of the first string and m is the length of the second string once it has been formed. The objective is to determine the value of the distance function *distance(n, m)*, which calculates the edit distance between the prefixes *x[0... n]* and *y[0... m]*. The distance function is computed as follows:

$$\begin{aligned} distance(n,m) = min(&distance(n,m-1)+1,\\ &distance(n-1,m)+1),\\ &distance(n-1,m-1)+cost(n,m)) \end{aligned} \tag{1}$$

If *x[n] = y[m]* then *cost(n, m) = 0*, otherwise *cost(n, m) = 1*.

Optimized Edit Distance with the bipartite graph: When we understand what an edit distance is, we can see that string-matching algorithms and edit distances both exhibit their accuracy and adaptability in the realm of music, and the results of earlier studies also support this result. However, they perform well only when comparing short pieces of music and perform poorly when it comes to situations such as transitions or swapping some pieces of music. The problem to be solved becomes difficult by comparing the similarity of long-duration songs, these songs can be judged as different even though they contain similar short segments. Thus, the calculation of local similarity is considered when we prepare the dataset because this will help improve the quality of string-matching algorithms such as edit distance.

Starting from that weakness and the analysis in the above edit distance algorithm. Tianyao He et al. [4] proposed a new theory for how the edit distance algorithm works. They treat a melody like a graph with consecutive notes, specifically an input piece of music will be broken down into smaller pieces respectively formulated into consecutive notes. And when two melodies are included for comparison, this forms a Bipartite Graph Matching. The graph perspective of melody similarity is illustrated in Figure 1.

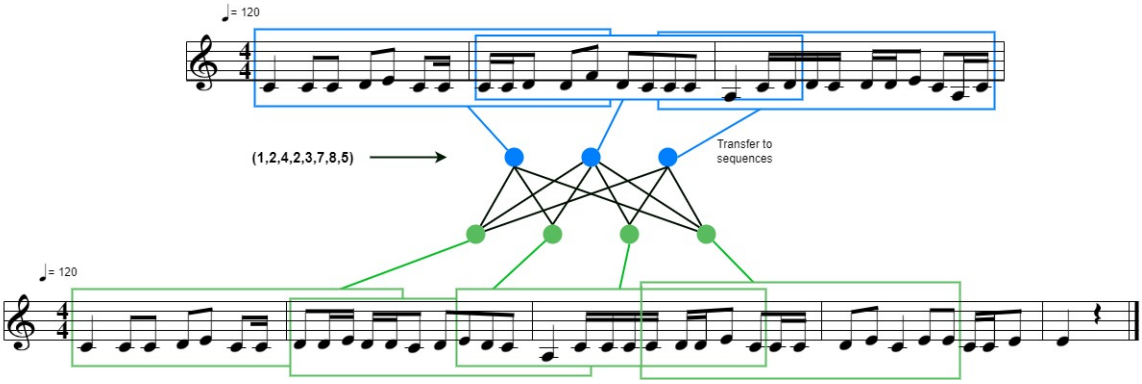

FIGURE 1. The graph perspective of melody similarity.

A familiar problem with bipartite graphs is the problem of finding the maximum weight matching for two sets of vertices of the graph. Specifically, the two matching sub-tone pieces have a high degree of similarity, if the maximum weight matching is larger, the two corresponding original melody pieces also have a high degree of similarity. Therefore, this problem is meant to calculate the similarity of two melodics. But it is necessary to calculate the individual weights of each edge first.

To deal with that problem, Tianyao He et al. [4] proposed a new way to convert musical notes when performing the conversion to "electronic melodies", specifically with pitch and duration representations to overcome editing limitations of the string matching algorithm. In addition, they proposed new meta-parameters to be included in the algorithm: consider pitch variation, duration variation, note downbeat as well as a consonance of the music, to improve accuracy and better suit music theory knowledge.

2.2.2. *N-grams.* N-grams are simply understood as a sequence of n consecutive characters appearing in a document sample [8]. In the field of music, N-grams have been applied as an approach to indexing audio-related data. And to enhance the model's accuracy, we apply some algorithms to measure the similarity and differences between the n-gram sequences extracted from the songs to be tested for plagiarism. A sample musical extraction is illustrated in Figure 2.

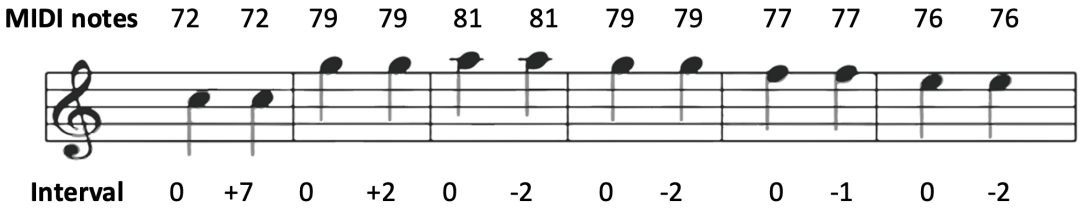

FIGURE 2. A sample musical extraction sheet indicates pitch intervals notation as (0, +7, 0, +2, 0, -2) which corresponds to the 2-gram will become (0, +7), (+7, 0), (0, +2), (+2, 0), (0, -2).

- **Sum Common**: This is a method to measure the similarity between strings and is calculated as follows:

$$SumCommon\,(s,t) = \sum_{\tau \epsilon A^s \cap S^t} \frac{f_{A^s}\,(\tau) + f_{A^t}\,(\tau)}{|A^s| + |A^t|} \tag{2}$$

  Where s and t are 2 musical sequences extracted from 2 different songs, A^s and A^t are the sets of distinct n-grams existing individually on s and t resp, $f_{A^s}\,(\tau)\,and f_{A^t}\,(\tau)$ represent are the n-gram τ frequencies in the series s and t respectively. The maximum frequency of an n-gram appearing in s is $s - n + 1$, so the maximum value of this calculation is $s + t - 2(n + 1)$.
- **Ukkonen Algorithm**: As an opposite measure to Sum Common, which computes the difference of non-simultaneous n-gram occurrences in both music sequences. The measure is calculated as follows:

$$Ukkonen\,(s,t) = 1 - \sum_{\tau \epsilon A^s \cap S^t} \frac{f_{A^s}\,(\tau) + f_{A^t}\,(\tau)}{|A^s| + |A^t|} \tag{3}$$

- **TF-IDF (Term Frequency – Inverse Document Frequency)**: According to the ability to measure the melodic element in a song, the TF part does not carry much meaning and would be omitted, but the IDF part will be focused.

$$IDF = \log\left(\frac{n}{n^\tau}\right) \tag{4}$$

  Where n is the number of strings in the dataset and n^τ is the number of strings τ in the dataset.
- **Tervsky Algorithm**; In this study, the computation of the Tervsky model is based on inheriting the result of the IDF measure.

$$Tervsky(A^s, A^t) = \frac{\sum_{\tau \epsilon A^s \cap A^t} IDF(\tau)}{\sum_{\tau \epsilon A^s \cap A^t} IDF(\tau) + \sum_{\tau \epsilon A^s \setminus A^t} IDF(\tau) + \sum_{\tau \epsilon A^t \setminus A^s} IDF(\tau)} \tag{5}$$

2.2.3. *Supervised Learning Algorithm (KNN and SVM).* Because of its straightforward implementation and efficient classification performance, KNN is an extensively used technique. The fundamental principle of a conventional KNN approach is to predict the label of a test data point using the majority rule, that is, by using the major class of the test data point's k most comparable training data points in the feature space [9]. Similar to other classification approaches, KNN classification is a two-step procedure that includes model training and test data prediction. The only step in the KNN training phase is determining a sufficient K for a specific training dataset [10]. The cross-validation approach is the most used one for this. A search for K data points in the training dataset that are most pertinent to a query (test data/sample) is the initial stage in the prediction phase [11]. The KNN will be taken by the most relevant points in the training dataset.

After that, the group of data that appears among the K neighbor most frequently will be used as a prediction.

Support vector machines (SVMs) are a collection of machine learning techniques that were first developed for the classification issue and then adapted to a variety of different contexts. The Vapnik-Chervonenkis (VC) dimension theory and the structural risk reduction principle are the foundations of the data learning technique known as SVM [12]. In this study, we apply Support Vector Classification. Having a collection of points that were labeled, this algorithm takes these data points and outputs the hyperplane (which is a line) that separates the tags. This line is the decision boundary which is used to classify points into tags. In SVM, an optimized hyperplane is the one that maximizes the margins from labels. We have N training points, where each input x_i has D attributes (i.e. is of dimensionality D) and is in one of two classes $y_i = 0$ or $y_i = 1$. Therefore, our dataset is of the form $\{x_i, y_i\}$ where $i = 1...L, y_i \epsilon \{0, 1\}, x \epsilon R^D$.

The hyperplane can be described by $wx + b = 0$ where w is normal the hyperplane and $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin. First, we need to create H where $H_{ij} = y_i y_j x_i \cdot x_j$ and find α so that $\sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha$ is maximized, subject to the constraints $\alpha_i \geq 0 \forall_i$ and $\sum_{i=1}^{L} \alpha_i y_i = 0$ This is done using a QP solver. Then we calculate w = $\sum_{i=1}^{L} \alpha_i y_i x_i$ and determine the set of Support Vectors S by finding the indices such that $\alpha_i < 0$. We continue to calculate:

$$b = \frac{1}{N_s} \sum_{s \epsilon S} (y_s - \sum_{m \epsilon S} \alpha_m y_m x_m \cdot x_s) \tag{6}$$

that each new point $x^{'}$ is classified by evaluation $y^{'} = sgn(w \cdot x^{'} + b)$. A kernel is an SVM function that facilitates problem-solving. It allows you to skip complicated computations by using shortcuts. With the assistance of linear classifiers, kernels are another method for solving nonlinear issues. Kernels like 'rbf' function well with smaller datasets. The likelihood of overfitting is increased by utilizing that kernel, though. For a better outcome, this study also uses a simpler kernel 'polynomial'.

3. **Methods.**

3.1. **Dataset.** Midi is a music format that stores 'event data' rather than sounds. These event data mostly contain note pitches, note velocities (loudness), instruments, and other information comparable to that found on sheet music. We collected a dataset of 46 cases, each case is an audio pair. The total is 92 songs. There are both verbal and non-verbal files. They will be converted to midi files by the melodic-segnet model of Hsieh et al [13]. Therefore, our dataset can meet the standard of input for all algorithms. The audio file as input after transformation by the model [13] will be used as a midi but with better quality than normal. The midi files no longer have noises that can affect the robustness of models.

3.2. **Representation.** All the approaches that we proposed require input files as midi files. As mentioned above, we have converted the audio dataset to midi for improved performance. On the other hand, local alignment shows that a rhyme can be a part considered plagiarism with other rhymes. As a result, only melodies that are suspected of plagiarism in each song in the dataset are used in order to increase the input element and improve models. After that, we also need different preprocessing data stages for each approach.

Before doing experiments, the input dataset of the editing distance and n-gram application have to be converted to the Numpy format, which is an array of strings. And as

for the supervised learning approach (KNN and Support Vector Machine), music21 (a python library) will be used to handle MIDI files in this case.

3.3. **Implementation & Evaluation methodology.**

3.3.1. *Edit distance and N-grams.* Regarding the basic edit distance, we only calculate the weights for the operations (insertion, deletion, or substitution) to compute the minimum number of operations for the two input melodies.

With a Bipartite Graph Matching graph, the algorithm will perform fragmentation of the original melody into pieces of length and overlap inherited from the previous study. After that, applied graph theory as analyzed above. The bipartite graph matching model will need to extract downbeat attributes during the evaluation process. In terms of n-gram, pitch and duration appearing on the piece length are two vital factors that need to be considered and selected randomly from the songs in the dataset. And the average index and the accuracy of the songs shown are two parameters that evaluate the effectiveness of these measurements.

By selecting a song from the dataset one by one and calculating the similarity of that song to all the rest. And if the song achieved the highest accuracy score is the song appearing in the same course, this means our model is correct. Therefore, "accuracy" means the average of the correct predictions over the total number of predictions times. Besides, "average index" means the average index row of the number of predictions. Specifically, when the results are returned, the correct song is ranked in the first line, which is the correct prediction. For the wrong case, the index row of the song in the pair compared with the inputted song will be on any line different than line 1. Thus, the "average index" will represent the degree (ability) of the algorithm to predict correctly.

And the results get more accurate when the returned "accuracy" is greater, while the opposite trend is true for the index.

3.3.2. *KNN and SVM.* In the machine learning method, we have two supervised learning models, which are KNN and SVM. First, to use KNN and SVM, collections of songs needed to be clustered. With a similarity score matrix, set a threshold for a song these scores, considering to what threshold a song can be concluded as plagiarism. Three thresholds—0.5, 0.6, and 0.7—have been defined for experimental reasons. Songs with similarity scores over the threshold will be flagged as potentially plagiarized. With similarity scores with every song in the dataset falling below the threshold, cluster #0 includes songs with no evidence of copying. On the other hand, cluster #1 contains includes songs that may be plagiarized and have a similarity score with any song more than the threshold.

4. **Experiment Results.**

4.1. **Edit distance.**

4.1.1. *Basic edit distance.* The algorithm has operations designed to match the 'addition', 'deletion', and 'substitution' of notes, making it immediately very promising. It is also a very simple and intuitive algorithm. We have designed the edit distance algorithm basically to get the best estimate of what a normal edit distance can do. However, from our experiences with the dataset, we obtain an accuracy of 46.4% and an average index of 2.875. This result shows a weak ability to detect songs that are similar to an original suspect song.

Because of the early well-known applications of edit distance applied in the field of natural language processing (NLP). So, it has limitations that need to be addressed. In some normal applications, the weight of the "operations" is considered constant. However,

in the musical context, this assumption must be discussed [14]. Because the effect of one pitch when replaced by another is not always the same. According to research by M. Mongeau et al [15], the substitution score may be correlated to the consonance interval. One problem with edit distance is that depending on the input string length, we don't have any rule about it. Therefore, how to choose the string length appropriately will be an aim of the research direction for this algorithm in the future. Besides, the problem of the shifts and permutations between pieces of music makes it difficult to detect similarities.

4.1.2. *Edit distance combine Graph theory (Bipartite Graph Matching).* The proposed algorithm can overcome the disadvantages of conventional edit distance, which are melodic transposition, note shifting, and card shuffle in plagiarism. Furthermore, the clever application of graph theory has further increased the accuracy when calculating the similarity. In particular, defragmenting the input sequences and treating them as nodes in the graph significantly improved the local alignment problem.

Therefore, we get an accuracy of 60% and an average index of 2.336. This result shows the potential of the algorithm with its improvements. The average index is quite small, which reveals that the number of times the index of the correct song is wrongly predicted with the suspect song is not much. Thus, the results can be trusted with our dataset.

In addition, this approach considers several types of pitch and duration representations. Pitch changes, duration variations, bass notes, and consonants are considered to calculate similarity. Overcoming the limitations of previous studies before ignoring the influence of these factors. Experimental results using basic and optimized editing distance measurements are shown in Table 1.

Table 1. Comparison result for String matching algorithms

Measure	Accuracy	Index
Basic Edit distance	0.46	2.875
Optimized Edit distance	0.60	2.336

4.2. **N-grams.** When choosing trigram and 4-grams [16], the experience returned a false positive in the entire data set, while that for detecting smaller scale cases was significantly more accurate. On the other hand, we also received some exceptions. Table 2 presents the results of the combination of N-grams and other measures.

Table 2. Results of N-gram combined other measures

Method	Accuracy	Index
TF-IDF Correlation	0.12	34.22
Sum Common	0.14	34.15
Ukkonen	0.13	29.18
Tversky Equal	0.13	34.55

Although there has been an improvement by combining N-grams with other algorithms, the result is only about 14%. The consequence can be explained that this technique concentrated on matching or mismatching when counting. Moreover, N-grams combined with other similar measures supposed that the number and frequency of N-gram sequences in the comparing process are governed by general perception-based correlation comparison, which may not be effective for small tracks of existing plagiarism.

In addition, the evaluation based on only 2 factors (pitch and duration) is indeed limited. Because a piece of music performed with different speeds or players will express different feelings. Therefore, each pitch performed independently is no longer the key, but it is the correlation with the other notes that should be taken into account.

4.3. **KNN and SVM.** Evaluation of classification by Machine Learning, we use values of precision score, recall score and accuracy. The more precise the prediction of points, the greater the precision score. Its beneficial characteristic is that it shows how many pertinent situations the +P rule detects. A good classification model is one that has both Precision and Recall high and as close to 1 as possible. The model results are the results after applying 5 folds. Experimental results for both methods are provided in Tables 3 and 4, respectively.

Table 3. Result of KNN method

Threshold for KNN	Accuracy	Precision	Recall
0.5	68.36	0.88	0.40
0.6	92.40	1	0.58
0.7	90.23	0.5	0.27

Table 4. Result of SVM method

Threshold for SVM	Kernel for SVM	Accuracy	Precision	Recall
0.5	rbf	80.35	0.79	0.79
	poly	67.37	0.75	0.46
0.6	rbf	89.24	1	0.43
	poly	87.08	0.9	0.37
0.7	rbf	89.18	0.2	0.1
	poly	92.46	0.8	0,4

Set three thresholds 0.5, 0.6, and 0.7 for songs to consider whether it is plagiarism or not. With threshold 0.5, the number of songs predicted to be in cluster "1" is more than other thresholds. The accuracy of KNN with this dataset at threshold 0.5 is 68.36%, not considered high. While the precision score is 0.88, considered high, the recall score is only 0.39. It means that this model with threshold 0.5 is not considered effective. Set a threshold at 0.6, the accuracy is 92.40%, the highest score in three thresholds. The model with this threshold is the most optimized model based on its results. The precision score is 1 and the recall score is 0.58, a quite good result. The threshold 0.7 has the accuracy is high, at 90.23%. However, the precision score and recall score of that method is deviant, bringing poor performance. In general, the implementation of two different kernels in training such as 'rbf' and 'poly' can be drawn that using one kernel 'rbf' is better than another. With threshold 0.5 for model SVM, the accuracy is 80.35% and both precision score and recall score is approximately 0.8. While the performance of kernel 'poly' is poor, three scores are quite low. The threshold 0.6 have performance is not optimized with both results of two kernels are not considered good. At kernel 'rbf', the accuracy score and precision score can be considered positive, but the recall is only 0.43. The performance of a model with threshold 0.7 is different from the others. In this model, the performance of kernel 'poly' can be considered as better than that of kernel 'rbf'. However, the scores of this model show that it is not an optimized model, since the recall

is low. From both performances, we can see that the KNN model with a threshold of 0.6 is the most optimized model. The best option for applying SVM is to use threshold 0.5 with kernel 'rbf'.

4.4. **Evaluation.** As analyzed in the experiment result section, we find that the experimental results with the Edit distance combine Graph theory method (Bipartite Graph Matching) are good and promising results. The algorithm first suggests using the MESMF method to convert physical musical notes into electronic formats more uniformly. In particular, they include giving musical notes additional attributes in addition to pitch and duration, such as consonance (which determines if a note is composed of resonances) and downbeat (which determines whether a note is a starting note or not). It computes the musical similarity quite efficiently. The model's capacity to identify plagiarism is the next intriguing feature. Additionally, the algorithm has a wide range of practical applications, which is consistent with the focus while designing commercials.
N-grams practices, which were adapted from these straightforward techniques consist of tallying the many elements that the query and probable results have in common. However, this method simply takes into account matching and mismatching when calculating the number of subsequences, seemingly oblivious to the fact that although musical fragments are comparable when taking into account music theory and perception, the two do not necessarily need to match exactly. Therefore, as demonstrated in our experiment portion, this strategy does not produce satisfactory results in experiments [4]. Therefore, in our experiment, n-gram features are combined with various similarity computation techniques, such as the Ukkonen measure, Sum Common measure, TF-IDF correlation, Tversky's measure, and so forth, which have demonstrated high performance. All of these measurements, however, are predicated on the idea that the quantity and frequency of similar or dissimilar n-gram elements are associated with the overall sense of similarity between two musical works. This approach is useless when dealing with instances where plagiarism is present, even to a very little extent.

For supervised learning algorithms, we conduct an analysis of TF-IDF and harmonic reduction first. Then we can retrieve insight into thresholds which is a feature in detecting music plagiarism. In both methods, the best performance is shown in threshold 0.6. This insight involves precision score and recalls score. These two scores have to be in proportion to each other to ensure the efficiency of the model, while the precision score is high does not mean the model is optimized. In addition, the accuracy score of KNN and SVM can reach 90% and 92%, as we can see in Tables 3 and 4. However, when we pick a song from group 1, meaning a group of possibly plagiarism songs, we can not know it is a copy of what song. Therefore, it does not have a significant affection on detecting music plagiarism in business. This algorithm is still effective for input preprocessing. In the future, we can reduce the original songs (which is not plagiarism) from the dataset for better performance. Moreover, we want to develop this method for comparing only two songs, we can put in the model. If the result shows that the similarity score of the two songs is greater than 0.6, we can assume that is a case of music plagiarism.

4.5. **Application.** We evaluate and select the algorithm that is most appropriate for study after putting the algorithms into practice. We have developed a tool for detecting music plagiarism with a user-friendly interface (see Figure 3). We initialize the application's system library including a dataset made up of 46 cases that have been gathered as mentioned in the dataset section. The user can choose an optional audio file that they suspect by clicking on the "Add music" button. The system allows users to select a song (mp3 or wav audio file format) in their own library. Frame (1) will display the song users have selected. At the same time, users can play, pause, and skip the song displayed in

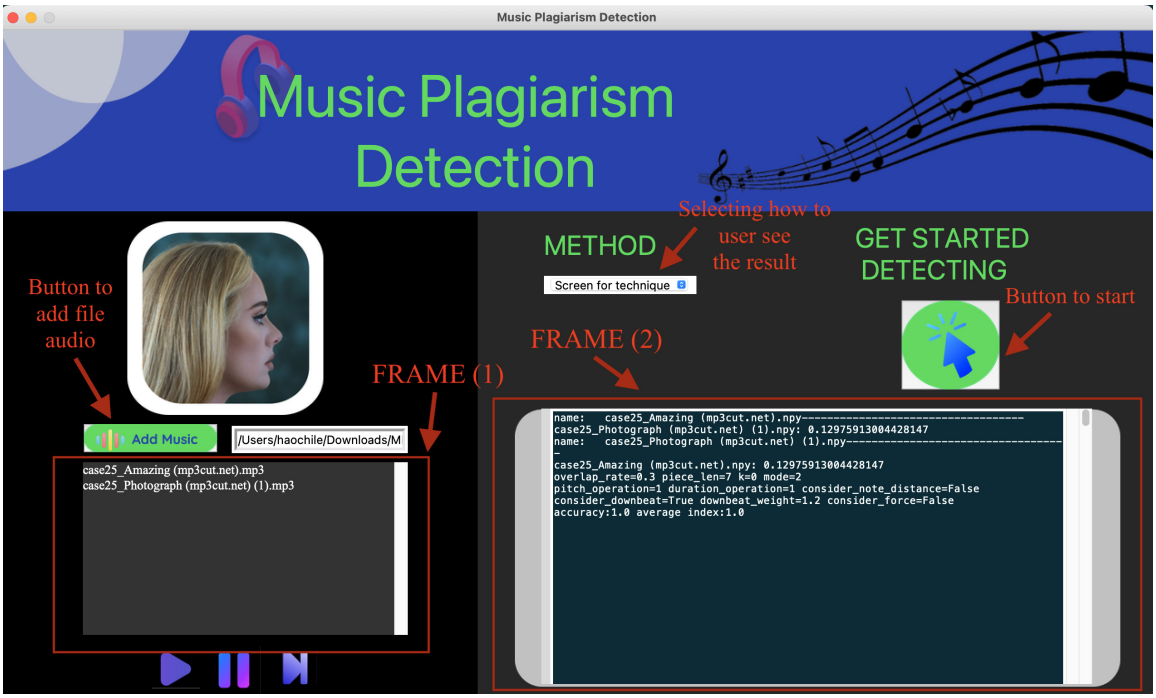

FIGURE 3. Our music plagiarism detection tool.

the frame. Then, in "METHOD", we build two selection methods that allow users to view the results in a frame (2): "Screen for techniques" - Screen is designed for users who want to see the information in technique view; "Explanation screen" - we designed an explained screen that is easy to understand for users with no technical knowledge.

The program will launch when you click "GET STARTED DETECTING". The following steps will be completed by the system: converting the audio file into midi format; extracting melodic features; running the model; and returning the outcomes in a frame (2). Additionally, users can also interact with the audio file that they have added and audio returned from the results. The result of the application will be displayed in the frame (2), the songs with the highest similarity score with the input songs will rank first, which means the higher the similarity score, the higher the ranking index of the songs.

5. **Conclusions.** Our research goal in this study is to detect similarities in the melody (music) using several methodologies that we have evaluated in prior studies. From there, new work for this study might be suggested. Specifically, we employ a supervised learning algorithm including K-Nearest Neighbor (KNN) and Support vector machine (SVM) (1); string matching algorithms include Edit distance (2) and Optimized Edit distance with Bipartite Graph Matching (3); N-gram (4). We also propose a new dataset to improve the objectivity of the implementation and to retest the current algorithms. We collect a dataset of melody recognition from real cases. Thus, through the analysis and prepared data sets, we evaluate that approach (3) is a good approach for this problem. Because it operates the music sequence by combining the benefits of both maximum weight matching and Edit distances. This overcomes many disadvantages of other methods as analyzed above. Prominent is solving the problem of shifts and changes in pitch and duration.

In the evaluation section, we make appropriate evaluations for each method. Our purpose is to find the algorithm properly for music plagiarism. Method (3) performs better than all other algorithms. This gives a potential insight into future development. Moreover, we split our dataset smaller and the results were quite good, the highest accuracy

was 0.875. According to this paper's result, there are other factors that affect this such as rhythm, and harmony, which we have ignored in this process. Because many musicologists and similar algorithms do not consider rhythm to be the determining factor in the similarity between songs [17, 18]. Moreover, these factors are significant and need to be studied further in the future.

In the future, we want to expand this research to study other features of melody so that we can improve and optimize the remaining limitations of this study. In addition, improving the input materials of the melody, especially in the remix or live music genre, is also a point of future development to increase the accuracy of the algorithm.

REFERENCES

[1] Schuitemaker, N. E. (2020). *An Analysis of Melodic Plagiarism Recognition using Musical Similarity Algorithms* (Bachelor's thesis).

[2] Robine, M., Hanna, P., Ferraro, P., & Allali, J. (2007, July). Adaptation of string matching algorithms for identification of near-duplicate music documents. In *Workshop on plagiarism analysis, authorship identification, and near-duplicate detection (PAN07)*, pp. 37–43.

[3] Dittmar, C., Hildebrand, K. F., Gärtner, D., Winges, M., Müller, F., & Aichroth, P. (2012, August). Audio forensics meets music information retrieval—a toolbox for inspection of music plagiarism. In *2012 Proceedings of the 20th European signal processing conference (EUSIPCO)*, pp. 1249–1253.

[4] He, T., Liu, W., Gong, C., Yan, J., & Zhang, N. (2021). Music Plagiarism Detection via Bipartite Graph Matching. *arXiv preprint arXiv:2107.09889.*

[5] Uitdenbogerd, A. L., & Zobel, J. (1998, September). Manipulation of music for melody matching. In *Proceedings of the sixth ACM international conference on Multimedia*, pp. 235–240.

[6] Rothstein, J. (1992). *MIDI: A comprehensive introduction* (Vol. 7). AR Editions, Inc.

[7] Lee, J., Park, S., Jo, S., & Yoo, C. D. (2011). Music plagiarism detection system. In *Proceedings of the 26th International Technical Conference on Circuits/Systems, Computers and Communications*, pp. 828–830.

[8] Heaps, H. S. (1978). *Information retrieval, computational and theoretical aspects.* Academic Press.

[9] Cheng, D., Zhang, S., Deng, Z., Zhu, Y., & Zong, M. (2014, December). kNN algorithm with data-driven k value. *In International Conference on Advanced Data Mining and Applications*, pp. 499–512.

[10] Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology (TIST), 8(3)*, pp. 1–19.

[11] Zhang, J., Qi, H., Ji, Y., Ren, Y., He, M., Su, M., & Cai, X. (2021). Nonlinear acoustic tomography for measuring the temperature and velocity fields by using the covariance matrix adaptation evolution strategy algorithm. *IEEE Transactions on Instrumentation and Measurement*, 71, pp. 1–14.

[12] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), pp. 273–297.

[13] Hsieh, T. H., Su, L., & Yang, Y. H. (2019, May). A streamlined encoder/decoder architecture for melody extraction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 156–160.

[14] Uitdenbogerd, A. L., Chattaraj, A., & Zobel, J. (2002). Music information retrieval: Past, present and future. *Current Research in Music Information Retrieval: Searching Audio, MIDI, and Notation. Kluwer.(originally presented at ISMIR 2000), to appear.*

[15] Mongeau, M., & Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities, 24*(3), pp. 161–175.

[16] Doraisamy, S. (2004). *Polyphonic music retrieval: The n-gram approach* (Doctoral dissertation, University of London).

[17] Müllensiefen, D., & Pendzich, M. (2009). Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicae Scientiae, 13*(1_suppl), pp. 257–295.

[18] Lemström, K., & Ukkonen, E. (2000, April). Including interval encoding into edit distance based music comparison and retrieval. In *Proc. AISB* , pp. 53–60.

[19] Park, K., Baek, S., Jeon, J., & Jeong, Y. S. (2022). Music Plagiarism Detection Based on Siamese CNN. *Human-centric Computing and Information Sciences*, pp. 12–38.

[20] Nair, R. R. (2021). Identification and Detection of Plagiarism in Music using Machine Learning Algorithms *(Doctoral dissertation, Dublin, National College of Ireland).*