Music Plagiarism Detection

based on Max-flow and Edit Distance

Chen Gong
Shanghai Jiao Tong University
gongchen@sjtu.edu.cn

Tianyao He
Shanghai Jiao Tong University
hetianyao@sjtu.edu.cn

Wenxuan Liu
Shanghai Jiao Tong University
wenxuanliu@sjtu.edu.cn

ABSTRACT

The revenue loss due to music plagiarism has been escalating exponentially. And today's timbre-based plagiarism detection algorithms may not be able to cope with the problems like transposition, tempo, swapping, shifting and so on. In this paper, we develop a novel music plagiarism detection algorithm based on max-flow and edit distance and we also take music theory into account to detect musically similar sequences. We also propose new experiment on several cases of music plagiarism judged by laws, where our algorithm shows high accuracy compared with other algorithms.

KEYWORDS

Music plagiarism, Max-flow, Edit distance, Experiment

1 INTRODUCTION

Today, the number of music documents available on the Internet is rapidly increasing. Each year, over 10,000 new albums of recorded music are released and over 100,000 new musical pieces are registered for copyright[7].

The revenue loss due to plagiarism and pirate copies has been escalating exponentially. With the development of music generation technology, machines are able to compose music if trained with enormous existing music data. How to avoid the plagiarism of generated music is extremely important.

Evaluating the similarities consists of computing a similarity measure between several pairs of music sequences. Some giant companies like Sony has attached their attention to this area. But the main of their algorithms are based on timbre similarity, mainly evaluated with relatively low-level features like cepstrum coefficients, which are hard for most of the people to understand. Also, their algorithms fail to cope with the problem of transposition which means the change of the major key, and the problem of tempo like the change of note duration. At last, the shifting and swapping problem also exist. It is possible that the whole song is a copy of another one, but the notes' order is shifted, and their algorithms cannot detect this situation.

In this paper, we present out new algorithms based on max-flow and edit distance. In Section 2, we describe how to compute the similarity between musical sequences. Then in Section 3, we show our procedure of maximum weight matching. In Section 4, we formalize the problem show our algorithms of **Representation of Music as Sequences based on Melodic features** (**RMS-M**) and calculating the **Similarity score based on Max flow and Edit distance**(**S-ME**). In Section 5, we demonstrate our improvement based on music theory. Then in Section 6, we show our algorithms and analysis its complex, and in Section 7, we show the results of our experiment.

2 SIMILARITY BETWEEN SEQUENCES

In order to compute the similarity between two sequences, we can firstly calculate their distance and then map the distance into a similarity score.

In Levenshtein' work [3], he defined the distance between two strings as the minimum cost of transforming one string into the other through elementary operations, which is also known as edit distance. The elementary operation mainly consists of deleting, inserting and substitution with respective costs denoted by c_d, c_i, c_s.

Supervised by Jiang Li.

Project in Computer Science (CS222), SEIEE, SJTU
2021.

According to work [8], the distance can be computed in time $O(|S_1||S_2|)$ using linear space based on dynamic programming algorithm where $|S_1|, |S_2|$ denote the length of two strings.

3 MAXIMUM WEIGHT MATCHING

In computer science, the maximum weight matching problem is the problem of finding, in a weighted graph $G = (V, E)$, a matching in which the sum of weights is maximized [9]. The traditional solutions to this problem are Hungarian Algorithm and Kuhn–Munkres algorithm(KM-Algorithm). KM-Algorithm takes time $O(n^3)$ to calculate the maximum weight of the matching where n denotes the cardinality of the matching.

4 PROBLEM FORMALIZATION

4.1 Problem Formulation

Given two songs A and B, we extract their melodic features and transform them into two sequences $A^s = \{a_1, \ldots, a_n\}$ and $B^s = \{b_1, \ldots, b_m\}$ where a_i denotes the i-th note of song A, b_i denotes the i-th note of song B and m, n denote the number of notes of two songs. Every note a_i is denoted as:

$$a_i = (pitch_{a_i}, duration_{a_i}, downbeat_{a_i})$$

which is established as a tuple with three elements where $pitch_{a_i}$ denotes its pitch feature, $duration_{a_i}$ denotes its duration feature and $downbeat_{a_i}$ denotes whether this note is a downbeat. Based on the melodic sequences, we are expected to compute the plagiarism degree $Similarity(A, B)$ of the two songs and further find all the pairs of pieces from two songs $A^s[start_A : end_A]$ and $B^s[start_B : end_B]$ which shows great plagiarism degree.

The algorithm based on the melodic similarity between two different songs consists of two parts. The first part is to extract the **Representation of Music as Sequences based on Melodic features** (**RMS-M**). The second part is to calculate the **Similarity** score $Similarity(A, B)$ between two melodic representations which reflects the plagiarism degree. The calculation of **Similarity score** is formulated as a max matching problem and can be solved **based on Max flow and Edit distance**(**S-ME**). This is a novel model to calculate the similarity between two sequences which has not been proposed in the related field. These two parts will be introduced in detail in the next two sections. The comprehensive procedure of our formalization can be understood clearly through Figure 1.

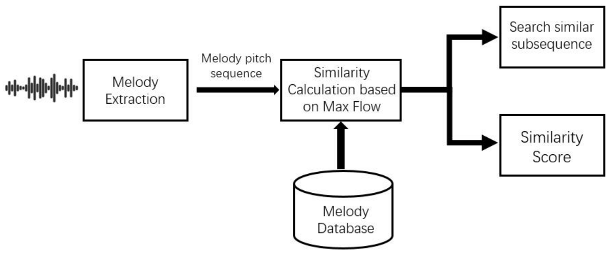

Figure 1: Procedure of Problem Formalization

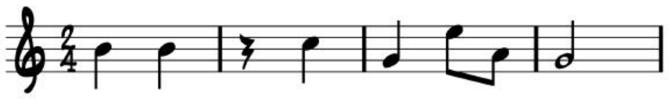

Figure 2: Example of A Musical Piece

4.2 Representation of Music as Sequences based on Melodic Features

According to Mongeau's work [4], each monophonic musical piece can be transformed into a sequence of ordered pairs by representing every note as a pair. In their model, a pair is composed of two parts: the pitch and duration of the corresponding note. For example, the melody shown in figure 2 can be transformed to the sequence:

$$(B4\ B4\ r4\ C4\ G4\ E2\ A2\ G8) \tag{1}$$

At the same time, work like [1] proposed different alphabets of characters and sets of numbers to represent both the durations and the pitches of notes . In this section, we show a few ones which we think are the most suitable for detecting music plagiarism.

4.2.1 Pitch Representation

To represent the pitches of notes of a music, there are three main methods: pitch contour, absolute pitch and relative pitch. All of them have their own advantages and we will give brief introductions to them separately.

(1) Pitch Contour:
The pitch contour describes the trend and variation between successive notes. For all the notes, they are represented by only three values: Up, Down and Same. The melody shown in figure 2 can be transformed to:

$$(S\ U\ D\ U\ D\ D) \tag{2}$$

The benefit of this method is that the range of value is really small which can largely reduce the time spent on calculating in the next procedure.

(2) Absolute Pitch:
The absolute pitch directly indicates the exact pitch of each note and we use the MIDI notation as standard. Therefore, the melody shown in figure 2 can be represented as:

$$(71\ 71\ 72\ 67\ 76\ 69\ 67) \tag{3}$$

To make the range of absolute pitch smaller and simplier, we change the exact pitches to modulo-12 values. Also, we can take the successive pitches' variation into consideration by assigning the note positive value when melody moves up and negative value otherwise. In this way, the melody of figure 2 can be represented as

$$(11\ 11\ \ +0\ \ -7\ \ +4\ \ -9\ \ -7) \tag{4}$$

The advantage of this method is that the sequence describes the pitch character of the music without information loss or neglect.

(3) Relative Pitch:
Compared with the previous two methods, relative pitch calculates the difference of successive notes(number of semitones). Then the melody in figure 2 can be represented as:

$$(0\ \ +1\ \ -5\ \ +9\ \ -7\ \ -2) \tag{5}$$

This method has strength in transposition invariation which means that increasing pitches of all the notes to the same degree will not influence the pitch sequence we extracted. This property has been proved to be meaningful and useful when detecting music plagiarism in our experiment.

4.2.2 Duration Representation

Similar to pitch representation, there are three main methods to represent the duration of each note: duration contour, absolute duration and relative duration.

(1) Duration Contour:
Similar to Pitch Contour, we use three values(Shorter, same, Longer) to describe the duration trend successive notes. Then the duration sequence of example melody Figure 2 is:

$$(s\ s\ s\ s\ S\ s\ L) \tag{6}$$

(2) Absolute Duration:
In terms of the absolute duration, we also use the duration defined by MIDI notation as standard which indicates the length of the note in sixteenth notes:

$$(4\ 4\ 4\ 4\ 4\ 2\ 2\ 8) \tag{7}$$

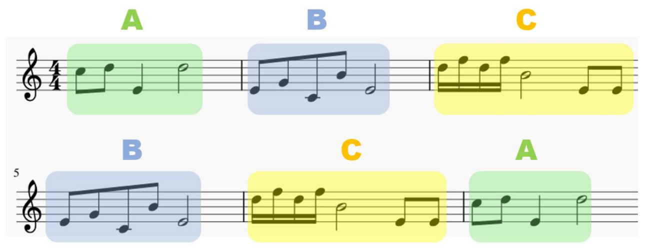

Figure 3: Example of Shifting and Swapping

(3) Relative Duration:
In order to make the sequence tempo invariant, we use the duration ratio to describe the difference of durations between successive notes:

$$(1\ 1\ 1\ 1\ \frac{1}{2}\ 1\ 4) \tag{8}$$

The tempo invariance means that speeding up or slowing down the whole music will not influence the duration sequence we extracted from the music.

4.3 Similarity based on Max Flow and Edit Distance

In the previous section, we transform two songs A and B into two sequences $A^s = \{a_1, \ldots, a_n\}$ and $B^s = \{b_1, \ldots, b_m\}$ by extracting the symbolic and melodic features. In this section, we want to compute the similarity score.

Some works like [5] directly formulate the similarity computation to a sequences comparison problem. They use edit-distance to obtain the minimum cost of transforming A^s to B^s. Their work shows good performance in short musical pieces comparison but shows really poor performance when considering cases like shifting or swapping some periods of a music. An example of this case is shown in Figure 3.

Some researchers [2] propose formulating the similarity computation as a local alignment problem. This method is theoretically better, but it also has not really good performance when experimenting on database because of less distinction about musical pieces. The results and analysis can be seen in section 7.

To solve the previous problems, we come up with formulating the similarity calculation between two music as a maximum weight matching problem. According to the melodic sequences extracted already, we establish a bipartite graph $G = (L \cup R, E)$ first and regard the value of the maximum

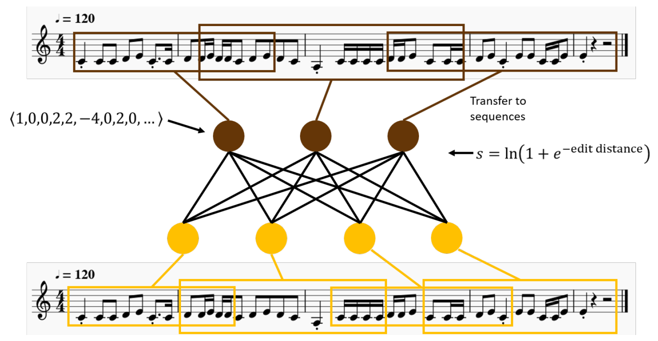

Figure 4: Example of Bipartite Graph Establishment

weight of matching as their similarity score. The establishment of the graph is introduced in the next section and figure 4 helps to understand this process.

4.3.1 Nodes Formalization

Firstly, we divide sequences A^s and B^s into pieces with the same length l and overlapping rate r which are two hyper-parameters and obtain two piece lists $A^l = \{A^s[0:l], A^s[(1-r)l : (2-r)l], \ldots\}$ and $B^l = \{B^s[0:l], B^s[(1-r)l : (2-r)l], \ldots\}$. The hyper-parameter overlapping rate r is used to avoid the situation that we cut an integral piece into different pieces and reduce our detection performance

Next, we formulate each piece in A^l as a node in the left node set L and each piece in B^l as a node in the right node set R.

4.3.2 Edges Formalization

For each node u_i in L and each node v_j in R, we construct an edge $e = (u_i, v_j)$ with value equaling to the similarity score between the two corresponding pieces A_i^l and B_j^l which means the i-th piece of A^l and the j-th piece of B^l.

The similarity score $Similarity(A_i^l, B_j^l)$ is computed based on the edit distance of these two sequences. Let us consider the three elementary operations that are usually used to compare musical sequences: substitution, insert and delete. Let e be an edit operation, a cost c is assigned to each edit operation as follows:

1) If e substitutes x_p (p-th character of A_i^l) into y_q (q-th character of B_j^l), then $c(e) = c(x_p, y_q)$
2) If e deletes x_p then $c(e) = c(x_p, \varnothing)$
3) If e inserts y_q then $c(e) = c(\varnothing, y_q)$

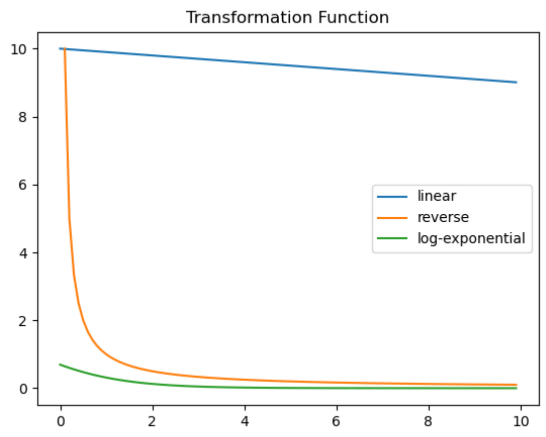

Figure 5: Curve of Three Transformation Functions

Then the edit-distance of sequences A_i^l and B_j^l can be calculated through dynamic programming:

$$\begin{aligned} d_{0,0} &= 0 \\ d_{p,q} &= \min \begin{cases} d_{p-1,q} + c(x_p, \varnothing) \\ d_{p,q-1} + c(\varnothing, y_q) \\ d_{p-1,q-1} + c(x_p, y_q) \end{cases} \\ d(A_i^l, B_j^l) &= d_{l,l} \end{aligned} \tag{9}$$

Initially, we assign each operation with the same constant cost. In the next section, we will use many tools to optimize the cost by taking music theory into consideration.

We only obtain the distance of two pieces and it is necessary to transform the distance into similarity score through transformation function $f : Similarity(A_i^l, B_j^l) = f[d(A_i^l, B_j^l)]$.

In terms of choosing function f, we have tried several different types of transformation functions like linear transformation $f_1(d) = \frac{100-d}{100}$, inverse transformation $f_2(d) = \frac{1}{d}$ and log-exponential transformation $f_3(d) = \ln(1 + e^{-d})$. Figure 5 illustrates the characteristics of the three transformation functions.

Through experimenting and comparing the results, we find that the log-exponential function has the best performance. This is reasonable because its slope decreases with the increasing of distance which means that when two pieces are different in a large degree, a little more distance will not influence the similarity score too much. Although inverse transformation function also has this property, it decreases too sharply when distance is small.

4.3.3 Solve the Problem based on KM-Algorithm

In previous sections, we have established a bipartite graph $G = (L \cup R, E)$ and then we can use the Kuhn–Munkres algorithm to compute the maximum weight of a matching and regard it as the similarity score of music A and B.

5 IMPROVEMENT BASED ON MUSIC THEORY

Substitution is the main edit operation which largely affects the performance of our music plagiarism algorithm. In our initial assumption, we assign every elementary operation the same constant cost. However, this assumption contradicts the real situation and ignores the music theory. For example, substituting a note ($pitch = 1, duration = 1, downeat = 0$) with a note ($pitch = 10, duration = 5, downbeat = 1$) affects the melody much more than with a note ($pitch = 2, duration = 1.5, downbeat = 0$). In order to improve the accuracy of our model, we take the music theory into consideration and come up with several optimization methods: considering pitch variation, duration variation, note downbeat as well as consonance of the music.

5.1 Pitch Variation

In most cases, the larger we change the pitch of a note, the more we will affect the original music. Therefore, we take the variation of pitch into consideration by directly using the difference of two pitches as the pitch cost of the operation:

$$c_{pitch}(a_i, b_j) = |a_i.pitch - b_i.pitch| \tag{10}$$

5.2 Duration Variation

When calculating the substitution cost between two notes, we can consider the variation of duration. Indeed, the insertion or deletion of a half note may disturb more significantly a melody than the insertion or deletion of a sixteenth note. Therefore, we use a hyper-parameter $k_{duration}$ to represent the relative importance of note duration:

$$c_{duration}(a_i, b_j) = k_{duration}|a_i, duration - b_j.duration| \tag{11}$$

5.3 Note Downbeat

According to music theory, there are strong beat positions and weak beat positions in a musical piece. The notes in the strong beat positions are of more significance to the music compared with the notes in the weak beat positions. We take this situation into consideration by giving the note in the strong beat position more substitution weight $k_{downbeat}(a_i, b_j)$ which is also a hyper-parameter:

$$c(a_i, b_j) = k_{downbeat}(a_i) \cdot k_{downbeat}(b_j) \cdot c(a_i, b_j) \tag{12}$$

5.4 Consonance

According to [4] and [6] work, the substitution cost may be correlated to the consonance interval. The substitution cost of note a_i and b_j based on the absolute pitch difference is determined according to the consonance: the fifth and the third major or minor are the most consonant intervals in Western Music. The associated scores are shown in table 1.

Difference	0	1	2	3	4	5	6	rest
Cost	0	5.7	5.325	3.675	3.675	2.85	4.65	3.35

Table 1: Substitution Cost according based on Absolute Pitch Difference according to Consonance

6 ALGORITHM COMPLEXITY ANALYSIS

In this section, we will analyze the complexity of our algorithm when calculating the similarity score of two music A and B. We suppose that A has n_1 notes, B has n_2 notes and we divide them into pieces with the same length l and overlapping rate r. Music A is divided into $\frac{n_1}{(1-r)l}$ pieces and music B is divided into $\frac{n_2}{(1-r)l}$ pieces and they need space $O(\frac{n_1 n_2}{(1-r)^2}) = O(n_1 n_2)$ to store these pieces.

To calculate the edit distance between two pieces with the same length l, we use dynamic programming whose time complexity is $O(l^2)$ and space complexity is $O(2)$.

To construct the bipartite graph, we need to calculate the similarity score between every piece from A and every piece from B which needs time $O(l^2 \cdot \frac{n_1}{(1-r)l} \frac{n_2}{(1-r)l}) = O(n_1 n_2)$ and space $O(\frac{n_1 n_2}{(1-r)^2 l^2}) = O(n_1 n_2)$. The graph has $\frac{n_1}{(1-r)l}$ nodes in the left set and $\frac{n_2}{(1-r)l}$ nodes in the right set.

Finally, we use the KM algorithm to compute the maximum weight of matching which needs time $O(\frac{n_1^3}{(1-r)^3 l^3}) = O(n_1^3)$ and space $O(n_1 n_2)$

To sum up, our algorithm needs time $O(n_1 n_2) + O(n_1^3)$ and space $O(n_1 n_2)$

7 EXPERIMENT EVALUATION

Due to the lack of well-accepted evaluation in the field of music plagiarism detection, we propose a new dataset and experiment method which can rationally reflect the ability of our detection algorithm.

7.1 Dataset

Our dataset is currently composed of 20 pairs of songs, where each pair is legally judged as plagiarism. We use the dataset from Ping An Tech as our baseline. The baseline contains 10 pairs of songs. Then, we extend it to 20 pairs by searching for lawsuits music plagiarism. We find the target songs from lawsuits, get their corresponding midi files, extract the melody and transform them to sequences (see Figure 6).

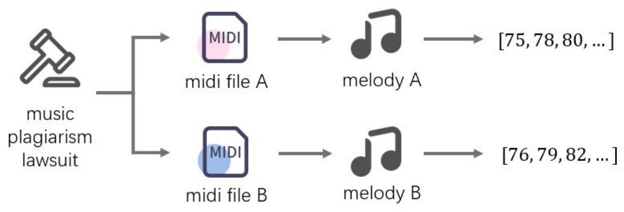

Figure 6: The process of our dataset establishment: finding music, getting midi files, extracting melody and transforming to sequences.

In this way, we get 20 pairs, in total 40 songs in the form of sequential arrays.

7.2 Experiment method

Our goal is to evaluate our algorithm's ability to find out the music plagiarism in the dataset and detect the similar sub-sequences. To do the experiment, We mix up the 40 sequences. Each time we pick out one of the song and calculate its similarity with other songs. If the song in the same pair (the plagiarized one) ranks first among all the other songs, we consider our algorithm as correct in this evaluation.

Finally, we can calculate the complete accuracy of the algorithm. The procedures are shown in Figure 7.

length	average index	accuracy
4	3.875	0.4375
5	3.1875	**0.625**
6	2	0.5625
7	**1.875**	**0.625**
8	**1.8125**	0.4375
9	3.75	0.4375
10	4.0625	0.4375
11	2.875	0.5
12	5.62	0.25
13	5.62	0.25
14	6.3125	0.25
15	3.0625	**0.625**
16	5.0625	0.5
17	5.5	0.4375
18	8.0625	0.1975

Table 2: average index and accuracy with different piece lengths

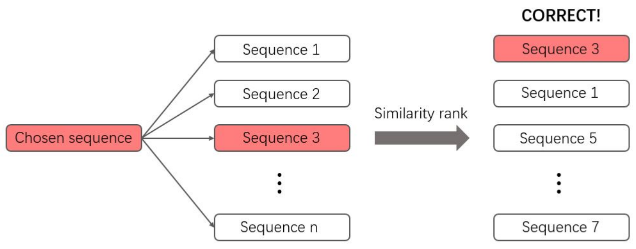

Figure 7: The explanation of our experiment

In the experiment, we focus on two factors: the average ranking index of the correct songs and the accuracy.

7.3 Evaluation Result

In the evaluation, we first tune three hyper-parameters: piece length, overlap and k. We first try different piece length(see Table 2), which means the number of notes.

The curves are shown in Figure 8.

We can find that the optimal piece length is 7. A short piece cannot well represent the melody of music, while a long piece may be not sensitive to similarity. 7 is close to about two to three bars in music, which can be the reason why it is optimal.

Then we tune the overlap rate (see Table 3).

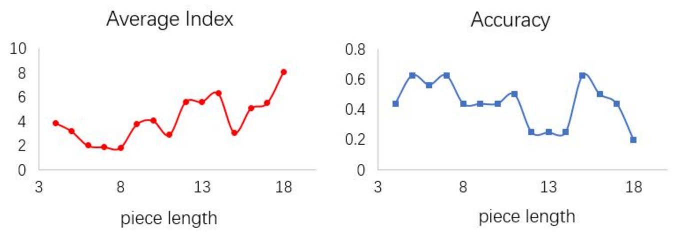

Figure 8: the average index and accuracy curve with different piece lengths

overlap	average index	accuracy
0.1	3.4375	0.4375
0.2	1.8125	0.625
0.3	1.875	0.625
0.5	2.5625	0.5625
0.6	2	0.5625
0.8	1.6875	0.625
0.9	**1.625**	**0.6875**

Table 3: average index and accuracy with different overlap rates

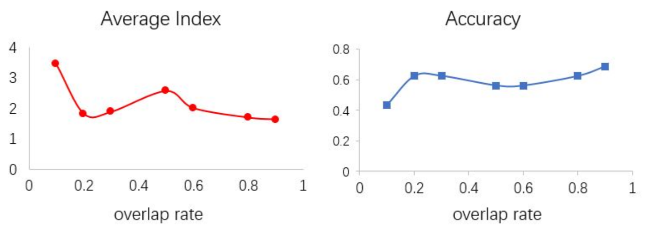

Figure 9: the average index and accuracy curve with different overlap rates

The curves are shown in Figure 9.

We find that 0.2 is the best overlap rate since it can best detect similarity and avoid meaningless repetition.

Finally, we tune value k (see Table 4).

Table 4 shows that $k = 0$ is optimal, which means the consideration of duration have negative effect on our algorithm. This may because the plagiarism often have little change on the duration. In this case, the relative duration may on the contrary make our algorithm confused.

Finally, we test different weights of downbeats in Table 5. The average index curve is shown in Figure 10. We get the optimal downbeat 1.5, which adds a proper importance to the downbeats since it affects the rhythm and emotion of music.

Finally, we compare our algorithm with other existing solution. The result is shown in Table 6. The baseline only using

k	average index	accuracy
0	**1.8125**	**0.625**
0.1	2.4375	0.4375
0.2	2.875	0.5
0.3	3.9375	0.4375
0.4	4.6875	0.4375
0.5	4.875	0.4375
0.6	5.25	0.4375
0.7	5.3125	0.4375
0.8	5.375	0.375

Table 4: average index and accuracy with different k

downbeat weight	average index	accuracy
1.2	2	0.625
1.3	1.9375	0.625
1.4	2	0.5625
1.5	1.875	0.625
1.6	**1.8125**	0.625
1.7	**1.8125**	0.625
1.8	1.9375	0.625
2	1.9375	0.625
2.5	2.0625	0.625
3	2.25	0.625

Table 5: average index and accuracy with different downbeat weights

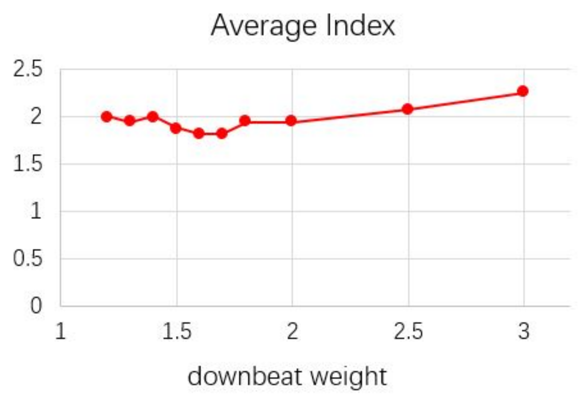

Figure 10: the average index curve with different downbeat weight.

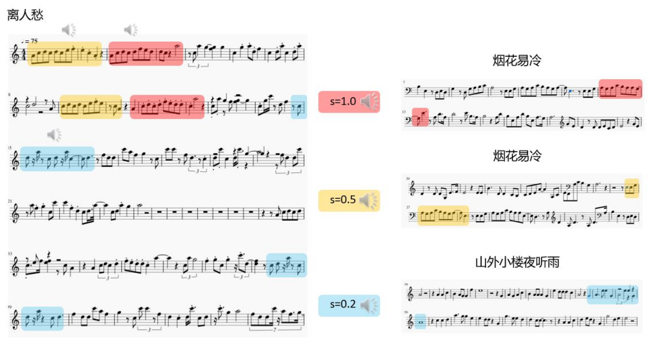

Figure 11: the demo of one-to-many similar pieces detection.

edit distance can only achieve 25% accuracy. With the help of max flow, our algorithm becomes robust to swapping and shifting and its accuracy achieves 56.25%. After we consider more music features such as relative pitch and note distance, our accuracy soar to 87.25% and the average index is 1.31. This shows the importance of major key and pitch difference in music.

Our algorithm is also capable of finding the similarity pieces of several music, which is our one-to-many detection. We here show a demo where we compare one famous Chinese song with other songs with potential plagiarism. The outcome is visualized in Figure 11. The same color means the similar pair of pieces. By listening the corresponding pieces, we find they actually has high auditory similarity. This also shows the power of our algorithm.

8 CONCLUSION

We proposed a new RMS-M sequential model to represent music in the form of sequence based on melodic features. We take pitch, major key, duration and downbeats into consideration. We also proposed a new similarity model S-ME to compute the music similarity. In this model, we combine the advantages of both edit distance and max-flow and apply it to music sequence. Our model is robust to sequential shifting and changes in key and duration.

In evaluation part, we designed our new evaluation dataset and experiment. The dataset consists of many pairs of music judged as plagiarism. Our experiment requires the algorithm to find the plagiarism in the dataset. Our algorithm outperformed all the other ones in the experiment.

In the future, we plan to extend our algorithm to music with multiple tracks and consider more auditory features to further increase our accuracy. We also plan to extend the application of our algorithm to other field, such as music search and music recommendation. Other important ideas such as collaborative filtering and neural network may also give us ideas to improve our algorithm. Finally, we want to improve our experiment and perfect our dateset to facilitate more researchers in this field.

REFERENCES

[1] J Stephen Downie. 2003. Music information retrieval. *Annual review of information science and technology* 37, 1 (2003), 295–340.

[2] Pierre Hanna, Pascal Ferraro, and Matthias Robine. 2007. On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences. *Journal of New Music Research* 36, 4 (2007), 267–279.

[3] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[4] Marcel Mongeau and David Sankoff. 1990. Comparison of musical sequences. *Computers and the Humanities* 24, 3 (1990), 161–175.

[5] Matthias Robine, Pierre Hanna, Pascal Ferraro, and Julien Allali. 2007. Adaptation of string matching algorithms for identification of near-duplicate music documents.

[6] Rainer Typke, Remco C Veltkamp, and Frans Wiering. 2004. Searching notated polyphonic music using transportation distances. In *Proceedings of the 12th annual ACM international conference on Multimedia*. 128–135.

[7] Alexandra Uitdenbogerd and Justin Zobel. 1999. Melodic matching techniques for large music databases. *Proceedings of the ACM International Multimedia Conference Exhibition*, 57–66. https://doi.org/10.1145/319463.319470

[8] Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)* 21, 1 (1974), 168–173.

[9] Wikipedia contributors. 2020. Maximum weight matching — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Maximum_weight_matching&oldid=966353106 [Online; accessed 8-January-2021].

Edit Distance	Direct Pitch	Relative Pitch	Max-Flow	Downbeat	Note-Distance	Index	Accuracy
√		√				6.50	0.2500
√		√	√			2.00	0.5625
√		√	√	√		1.88	0.6250
√		√	√		√	**1.31**	**0.8750**
√		√	√	√	√	1.38	0.8125
√	√		√			4.50	0.5625
√	√		√	√		4.00	0.5625

Table 6: average index and accuracy with different downbeat weights.