

An Analysis of Melodic Plagiarism Recognition using Musical Similarity Algorithms

Nick Schuitemaker (n.e.schuitemaker@students.uu.nl)

College number: 6259855

Bachelor Kunstmatige Intelligentie, UU

June 2020 - 7.5 ECTS

Attendant 1: Frans Adriaans

Attendant 2: Jakub Dotlacil

Abstract

Court cases on melody infringement are decided based on vague unspecified terms like the ‘substantial similarities’ between melodies. It is then up to experts to unbiasedly explain these concepts to an untrained judge, which has often lead to controversial verdicts. This paper presents an attempt at objectifying vague terms like ‘substantial similarity’ in cases of melodic plagiarism by testing whether existing melodic similarity algorithms can help determine plagiarism. Perceived similarity by ordinary humans (intrinsic test) and expert analyses in court (extrinsic test) are used to gather requirements that an algorithm needs to meet in order to accurately predict plagiarism. Some well-known similarity algorithms and pre-processing algorithms are presented and analyzed on whether they meet the requirements. The Edit Distance algorithm does well on these requirements and is then used to compare a song to both remixes of that song and non-remixes, finding an optimal boundary between the two that we can label as a plagiarism boundary. Lastly, we discuss the performance of the algorithm and examine the potential of this approach.

Introduction

Recently, Katy Perry was sued for 2.8 million dollars over a lawsuit claiming her song Dark Horse was similar to the song Joyful Noise from Marcus Grey. This incident sparked a debate as musicians became confused about the requirements for plagiarism. The law is not very clear on this matter. For melodic copyright, the law says that a song is considered plagiarized if the defendant had ‘access’ to the copyrighted song, and if there is a ‘substantial similarity’ between the two melodies. The former criterion is often decided based on the popularity of a song and is not of importance in this paper. The latter criterion, however, is of great importance, as it is a problem that there are no rules on how two melodies are to be compared, or when a substantial similarity is reached.

In practice, expert musicologists are requested to explain how similar the two melodies really are. The result of this is that ‘disputes would devolve to battles between experts, cherry-picking commonplace unprotectable similarities between two works in an attempt to manipulate musically untrained juries into findings of substantial musical similarity’ (Pettersson, 2019), and as shown in the case of Katy Perry, the verdicts can be very controversial. After revisiting the case, the judge claimed that the copyrighted material was too simple and common to be called plagiarized, and reversed the verdict. (Pettersson, 2019)

What we can see from this is that the rules are vague and often inconsistent, and this is a significant problem. Companies can lose millions of dollars in a lawsuit that is based on vague inconsistent rules like this. To solve this problem, we attempt to find an objective solution to help measure plagiarism. Many algorithms have been proposed to determine similarities between songs. Mainly, these are so-called ‘Fingerprinting, algorithms designed to recover a song based on a possibly inaccurate sample. We intend to seek out the feasibility of using these algorithms to determine plagiarism in music. This contributes to the area of Artificial Intelligence by finding important use for computational methods to aid important legal decisions.

Because these algorithms were never intended for this purpose, one has to be selected based on several attributes. The selected algorithm will then be tested to see how well it can distinguish between remixes of the song in question and non-remixes. To explain this decision: remixes, next to covers, are considered boundary cases for copyright, in that they are largely original but contain just enough copyrighted material to be considered plagiarism without the correct credits. (Casey, 2007) Expected is that a correct similarity algorithm will notice the similarities in melody between a song and its remixes. It is also expected that the algorithm notices the significant differences in melody between a song and other unrelated songs. Therefore, an algorithm being able to separate remixes from different ordinary music is a decent way of spotting its ability to measure similarity for plagiarism cases. On top of this, it will also show where such an optimal threshold would lie.

As such, the goal of this paper is to assess the applicability of (a variety of) musical similarity measures in defining a threshold for melody plagiarism cases.

In order to accomplish this goal, several questions require answering. Firstly, we need to know what defines a melody. The way humans perceive melodies is not obvious, and different people perceive melodies differently. It is also not obvious how we perceive similarities between different melodies. For example, many musicologists and similarity algorithms do not consider rhythm a factor in deciding similarity between songs. (Neely, 2020) (Müllensiefen, 2009) (Ukkonen, 2000)

Secondly, we need to know what defines a proper similarity measure for plagiarism. There exist many similarity

algorithms, all functioning slightly differently. The question that arises is how can we use our knowledge of melody perception to choose which algorithm predicts plagiarism the best.

The structure of the paper is as follows. Firstly, the current rules for melodic plagiarism and what these entail for the requirements of similarity algorithms will be reviewed. Secondly, several similarity algorithms are discussed on how well they satisfy the requirements for finding plagiarism are discussed. Thirdly, the chosen algorithm is tested for optimal separation of remixes and non-remixes. Then, the results are presented. We conclude with the explanations and implications of the results.

Literature

Many similarity algorithms have been proposed for 'Fingerprinting' systems. Some are designed to work with audio signals (Casey, 2007), with sheet music (Foscarin, 2019), and with MIDI files (Ukkonen, 2000). MIDI formats are an obvious choice here, since other formats like video are often noisy and can change a lot with each performance. It also requires a lot of time to separate all the instruments to get the melody. MIDI takes care of all these issues by presenting the song in a clear ordered structure. For more information on what a Midi file looks like, see the 'Methods' section.

To find the melody from a MIDI track is still not an easy task. Several algorithms have been proposed to work with both Polyphonic music (Uitdenbogerd, 2002), where the melody is divided between different tracks or instruments, and Monophonic music (Ukkonen, 2000) (Müllensiefen, 2009) (Rigaux, 2019), where the melody is contained in a single track. Given that we often deal with Pop music, the choice for a Monophonic system is justified. (de León; Antonio Pertusa; Carlos Pérez-Sancho; José Manuel Iñesta Quereda, 2006) For these forms of music, only the track containing the melody has to be found, and for this there have been proposed algorithms. (de León; Antonio Pertusa; Carlos Pérez-Sancho; José Manuel Iñesta Quereda, 2006) However, since these algorithms often require training or are inaccurate and difficult to set up, we will simply loop over all tracks in a song and keep the best performing track.

Melodic Plagiarism

There are many factors that play a role in deciding what a similarity algorithm must do in order to correctly label plagiarism. Here we will analyse some of these factors.

Court Decisions

Firstly, we will discuss how court decisions are made. In order to prove that your melody has been plagiarized, two things need to be shown.

The first thing to show is that the defendant had 'access' to your melody, meaning that the defendant has heard the

melody before. A song is considered plagiarized if you consciously or unconsciously stole pieces of the song, and thus you cannot have stolen it if you have never heard the song in question. This is often proven with circumstantial evidence, for example by viewing the popularity of a song by the view-count on sites like Youtube.

The second thing to show is that the melodies are 'substantially similar'. What this entails however is not defined clearly. In order to test for substantial similarities, courts often use different kinds of tests. Two commonly used ones are the extrinsic test and the intrinsic test.

The Extrinsic Test

The extrinsic test relies on expert musicologists explaining all the elements that are similar between the songs. A lot of factors are considered, like chord progressions (harmony), lyrics, the overall structure of the songs, rhythm, and of course the actual note-sequences. (Stav, 2014)

We will go over some of the most important factors here.

Harmony Especially in pop music, a melody is often harmonized, meaning it has an underlying chord progression, a chord being multiple notes played at once. Though a copyright claim is never solely based on these chord progressions alone, they can enhance the feeling of similarity and are thus considered in similarity analyses. (Pinter, 2015) However, chord progressions are not an intrinsic part of each melody and can be considered separate from the melody, which is why harmony will not be looked at in this paper.

Rhythm Rhythm is a combination of the timing and duration of notes. Whether rhythm is an important factor in deciding similarity is doubtful.

Rhythm proves to be of at least decent importance in court cases (Stav, 2014), and a melody is often considered to be only a combination of pitch and rhythm, suggesting that these factors are of equal importance. (Pinter, 2015)

On the other hand, in trying to brute-force all commonly-used melodies, expert musicologist Damien Riehl and his collaborator Noah Rubin purposely chose to ignore rhythm, stating that rhythm is not an important factor in perceiving similarity between melodies. (Neely, 2020) Also, many similarity algorithms, including most of the algorithms we will be considering here, typically ignore the rhythmic aspect of a melody completely. However, some papers that have previously not incorporated rhythm in their algorithms have left it open for intended future work. (Müllensiefen, 2009)

Preferably, a similarity algorithm is invariant to a decent amount of rhythmic variations, yet still considers rhythm in its decision. (Rigaux, 2019) But given the lack of existing algorithms incorporating rhythm, we will mostly ignore rhythm as a factor in deciding an algorithm.

The Intrinsic Test

The intrinsic test, also called the 'ordinary observer test', is one in which an ordinary person explains whether the two melodies in question sound substantially similar to them.

Note sequences An attempt to formalize and model this test is difficult, but attempts have been made. In 2006, an experiment on tune recognition concluded that, on average, 6 consecutive notes are needed for ordinary persons to recognize a tune. (Dingfelder, 2006) This can suggest a plagiarism boundary of 6 identical consecutive notes.

There seems to be a common conception that a certain amount of identical consecutive notes is important to the decision of plagiarism: there seems to be an unwritten rule stating that approximately five to nine consecutive notes the minimum threshold is for plagiarism (Pinter, 2015); Stephen Schwartz, composer of the musical 'Wicked', once stole 7 identical notes from another song believing that 8 notes is the plagiarism boundary (Giere, z.d.); attempts at brute-forcing most common melodies chose the limit of 12 identical notes, hoping to incorporate most melodies with it. (Neely, 2020)

There is some truth in these numbers, given that for every added note in a melody, the chances of accidentally coming up with that same melody decreases exponentially. (Pinter, 2015) However, actual similarity analyses are based on many more factors, including harmony and rhythm as we have discussed before. (Stav, 2014) Still, keeping the length of the melody in mind is certainly important to keep crucial elements of music in the free domain.

Common note-sequences Even two melodies that are long in length and match in both pitch and rhythm cannot always be considered plagiarized. (Pinter, 2015)

Similarly to how some chord progressions are staple in pop music, which many parody songs show¹², so are some melodic lines. When trying to create a catchy tune, many melodies simply do not sound good enough and the amount of possible combinations drop significantly. (Stav, 2014)

A clear example is the appeal for the lawsuit against Katy Perry's song 'Dark Horse', where the judge suggested that 'the eight-note section of Dark Horse in question was not a particularly unique or rare combination of notes.' It was also suggested that the opposing side was 'trying to own basic building blocks of music, the alphabet of music that should be available to everyone.' (Beaumont-Thomas, 2020)

Such an argument, suggesting that material is too common and simple to be claimed as one own, is referred to as 'Scènes à faire', and is an important rule that keeps building blocks of music in the free domain. As such, we will consider it in our analysis.

Independent Creation Related to this topic is the topic of 'independent creation', representing the possibility of two songwriters independently coming up with the same melody.

One example of this happening is with the song 'All About That Bass' of Meghan Trainor which is almost identical in pitch and rhythm as a Hungarian school-camp song named 'Napkorong Az Égről'. It is very unlikely that Meghan took inspiration from this song in the making of her own, and thus

no song can be considered plagiarized here. (Pinter, 2015)

Though no articles have yet been able to give strong conclusions to support independent creation, there have been articles showing promising evidence pointing at its direction. (Frieler, 2009)

This points out something important. Longer note-sequences are less likely to be similar between songs, common note sequences are more likely to be similar between songs, and since independent creation is plausible, we have a strong reason to keep the two previously mentioned factors in mind when determining plagiarism.

Melodic Alterations

Simple changes can be made to melodies without altering the perceived similarity. Thus an algorithm should keep these changes in mind.

Changing Key Changing key (or: 'transposing') is perhaps the most simple way to alter a song without significantly changing its content. First we will give some background information.

A scale is a set of intervals. A key has the name of a note, called the 'root', and is the set of notes obtained from applying the intervals of the given scale to the root note. An example would be the key of A Major, which applies the Major scale, consisting of the intervals (2, 2, 1, 2, 2, 2, 1), to the note A, resulting in the notes A, B, C♯, D, E, F♯, G♯. A 'sharp' (♯) or 'flat' (♭) referring to the first note directly right and left from the note named before.

Changing the key is done by simply changing the root note. In this way, the song be played a little bit higher or lower, but it is still considered and perceived as the exact same song since the notes in a song are heard as relative to their key.

Changing Mode Other than changing key, you can also change a mode, which, for the purpose of this paper, can be seen as identical to a scale. We have already seen the intervals of the Major Scale. The most common modes are Major and Minor. These two modes are insignificantly different from each other, and thus changing mode does not typically change a melody from being plagiarized to being original. For this reason, Riehl and Rubin considered both major and minor variants of the melodies in their effort of creating a database of all commonly used melodies. (Neely, 2020)

Remixing When remixing a song, it is typical to add your own spin on it. This can be done by altering existing parts ('substitution'), or by adding ('addition') or removing ('deletion') certain elements. One example we have already shown of an insignificant substitution was in changing the mode. Thus, we will consider a change in mode as a substitution.

These three alterations (substitution, addition and deletion) are very important to similarity algorithms, since they make sure that simple and common changes like a single added note do not change the output of the algorithm significantly.

¹<https://www.youtube.com/watch?v=o01DewpCfZQ>

²<https://www.youtube.com/watch?v=JdxkVQy7QLM>

Concluding Factors

Many factors go into deciding whether a melody is plagiarized. The most important factors are

1. **Changing the key.**
2. **Addition of notes.**
3. **Deletion of notes.**
4. **Substitution of notes.**
5. **Commonness of note progressions.**
6. **Length of the sample.**
7. **Rhythm.**

Rhythm is sadly considered by barely any algorithm. It is also considered a far less important factor than the actual melody, to the point where some musicologists doubt its relevance at all. (Neely, 2020) For this reason, we consider rhythm less relevant in deciding the algorithm.

Preferably, an algorithm keeps all of these factors into consideration when calculating the similarity. If not, experts choosing to use an algorithm to support their case have to keep some of these factors into consideration themselves.

Similarity Algorithms

Many algorithms have been proposed for finding melodic similarities. We will be going over a selection of some of the most common or promising ones. The algorithms selected are as follows:

1. **Edit Distance**
2. **Tversky.plaintiff.only**
3. **The Comparative Method**

We will go over these algorithms and discuss how well they meet the requirements mentioned in the previous paragraph. First, however, we will take a look at some pre-processing steps we can perform to develop a useful representation.

Pre-processing Steps

In order to allow the algorithms to work well, we need to feed them the data in the right format. Many of the requirements we have discussed are easily dealt with before selecting an algorithm. We will discuss the development of a correct representation here.

Intervals One problem that was mentioned was that of changing keys. The notes are heard relative to the previously played notes, giving the feeling of a song being played inside a key. Changing the key of a song changes the absolute sounds, but not our perception of the sounds.

A simple and common solution to this problem is to encode the intervals between notes instead of the notes themselves. Then, it does not matter whether all the notes are scaled up or down, because we look at the distances between them.

The use of intervals does bring some complications however. Consider a sequence as follows:

(70, 70, 75, 80, 80).

If we translate the sequence to its interval-variant, it would look as follows:

(0, 5, 5, 0).

What could be problem here is that the representations differ significantly in their values. Originally having a repetition of notes, (70, 70, 75), is likely to lose that feature, (0, 5). Also, an algorithm like Edit Distance will now add 2 to the similarity value for each 1 note altered, since a single altered note changes both the interval *to* this note as the interval *from* this note. This is something to keep in mind.

Chords The harmonic aspect of a song typically refers to the underlying chord progression. Here, we will not be talking about this aspect of harmony, but rather about obtaining the melody line when it is contained inside a chord progression. A melody is typically represented as a sequence of single notes, while a song contains a large amount of instruments and chords. Thus, deriving the melody from a song can be difficult.

When extracting a melody, there will always be inconsistencies. The perception of a melody is subjective, and it can change a lot per person and from time to time. This is partly due to the ability to focus on different parts of a song. Switching our attention in this way can allow us to hear different melodies in a song. (Rigaux, 2019) (Uitdenbogerd, 2002)

However, research has been done on the topic of perceived melodies, and typically in the case of chords, we perceive the top note of a chord the strongest. (Uitdenbogerd, 2002) Therefore, in the case of chords, we pick only the top note for our melody.

Rhythm As we have discussed before, most algorithms typically ignore rhythm as a whole. To get rid of all rhythmic material, the pre-processing algorithm 'Vnorm' has been suggested. (Rigaux, 2019) This algorithm deletes rests, merges repeated notes (which we consider as an elongation of a single note), and makes all notes of equal length. The last step is not needed in our case, as we ignore the length of notes altogether.

Chosen algorithm

Edit Distance This is a very simple algorithm, originally designed as a string matching algorithm, which measures the minimal amount of 'operations' one needs to perform to change one sequence into another. It does this by going over the possible permutations and taking the path that requires the least permutations.

Different sets of operations can be used, and you can pick or define your own set that would fit the task at hand. Most often, the Levenshtein distance operations are used, consisting of 'insertion', 'deletion', and 'substitution'. These operations are applied to individual characters, and they all have a weight attached to them. When using the Levenshtein distance operations, most often the weights are all set to 1. Formally defined, the following operations are used:

1. **Insertion** of a single symbol: this is denoted as $(\lambda \rightarrow a)$ for some symbol a , λ representing an empty symbol. The weights for this operation are $w(\lambda \rightarrow a) = 1$ for all a .
2. **Deletion** of a single symbol: this is denoted as $(a \rightarrow \lambda)$ for some a . The weights for this operation are $w(a \rightarrow \lambda) = 1$ for all a .
3. **Substitution** of a single symbol: this is denoted as $(a \rightarrow a)$ or $(a \rightarrow b)$ if $a \neq b$. The weights for this operation are $w(a \rightarrow a) = 0$ for all a and $w(a \rightarrow b) = 1$ for all $a \neq b$.

The minimum amount of operations needed is calculated efficiently. First a matrix d is made of size $n \times m$, n representing the length of the first sequence, and m representing the length of the second sequence. The goal is to find $d_{n,m}$. There exists a well-known recurrence to calculate the values of this matrix efficiently without trying all possible paths. This recurrence is as follows:

$$d_{0,0} = 0$$

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} + (\text{if } a_i = b_j \text{ then } 0 \text{ else } 1) \end{cases}$$

Other commonly used sets of operations include the Longest Common Subsequence operations, which does not allow for substitution (except self-substitution), or the Hamming Distance operations, which allows only substitution. Since both addition, deletion and substitution are factors we consider, we will use the Levenshtein distance operations.

An example of the Edit Distance algorithm at work is shown in Figure 1.

Evaluation 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.

The algorithm does not keep in mind the commonness of note progressions, nor the length of the sample. The algorithm also does not keep rhythm in mind, but as noted before, this might not be as important as the other factors.

Other algorithms

Tversky.plaintiff.only This algorithm uses a concept called 'n-grams', which are simply sequences of n items. An example would be a 2-gram containing 2 pitch intervals.

This specific algorithm is based on the idea that human similarity perception is a combination of the amount of features in common between two objects and the salience of these features. (Müllensiefen, 2009) Tversky's algorithm is originally a set-theoretical algorithm, but it can be combined with another n-gram algorithm called 'TF-IDF' to be used for musical purposes. (Müllensiefen, 2009)

'TF' and 'IDF' are separate parts of a combined algorithm, both operating in a similar fashion. First, the sequence of

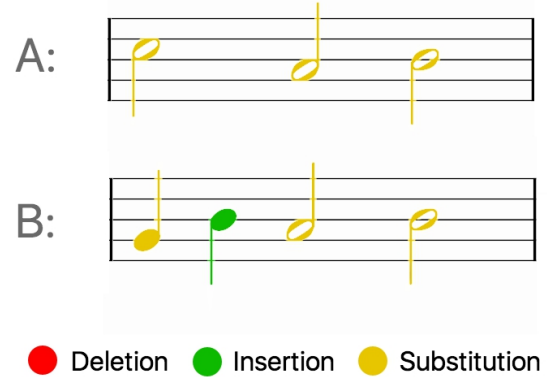


Figure 1: The Edit Distance algorithm is run with the top melody (A) and bottom melody (B). The last two notes of A are being Substituted for themselves ($w(a \rightarrow a) = 0$). The first note of A is Substituted for another note ($w(a \rightarrow b) = 1$), and the second note of B is Added ($w(\lambda \rightarrow a) = 1$). Thus, the similarity value of these melodies is 2.

items (pitch intervals in our case) is split up into all possible n-grams, meaning: in all possible sets of n consecutive items. As an example, consider a sequence like '(+0, +5, -3, +1)'. All possible 2-grams of this sequence would be as follows: '(+0, +5), (+5, -3), (-3, +1)'.

Next, each of these n-grams, called 'terms' (denoted by τ), are separately used in the algorithm. The 'TF' part is not of importance here, given that this algorithm does not use it. The 'IDF' part tries to measure how often a term occurs in other songs from a given database. The formula is as follows:

$$IDF_C(\tau) = \log\left(\frac{|C|}{|m : \tau \in m|}\right)$$

with $|C|$ denoting the amount of melodies in the database C , and $|m : \tau \in m|$ denoting the amount of melodies in C that contain the term τ at least once.

Intuitively, this score gets higher as the amount of songs containing the term decrease. Thus, it represents how uncommon the term is.

Tversky's model has the form:

$$\sigma(s,t) = \frac{f(s_n \cap t_n)}{f(s_n \cap t_n) + \alpha f(s_n \setminus t_n) + \beta f(t_n \setminus s_n)}, \alpha, \beta \geq 0$$

with f representing some form of 'salience function', which is related to intensity, frequency, or other factors that influence perceptual salience. Here, the IDF function is used as the salience function.

The weights α and β are to be decided based on how similarity is computed.

An α of 1 and a β of 0 looks only at terms in the first melody that are not contained in the second melody, and not the other way around. An α of 0 and a β of 1 does the opposite. Which choice you make depends on if you look at whether the defendant’s melody matches all the material of the plaintiff’s melody, or if you look at whether the plaintiff’s melody matches all the material of the defendant’s melody. Both methods are found in expert analyses.

In (Müllensiefen, 2009), the plaintiff.only version worked best out of all algorithms considered, including algorithms like the Edit Distance. The plaintiff.only version is one with α set to 1 and β set to 0, meaning that all features of the plaintiff are matched against the melody of the defendant, but not the other way around. Because of these promising results, this is the variant we will be considering here. Combining everything, the formula now looks as follows:

$$\sigma(s, t) = \frac{\sum_{\tau \in s_n \cap t_n} IDF_C(\tau)}{\sum_{\tau \in s_n \cap t_n} IDF_C(\tau) + \sum_{\tau \in s_n \setminus t_n} IDF_C(\tau)}$$

Because of its complexity in look and calculation, it is harder to visualise than the other algorithms. We do attempt to show an example of the Tversky.plaintiff.only algorithm at work in Figure 2.

Evaluation Adding, deleting and substituting notes brings more complex changes to the outcome of this algorithm than that of the Edit Distance, whose operations were designed for these purposes specifically. Intuitively, two melodies having all notes in common gives a similarity value of 1, and every note you change, either by adding, deleting or substituting, will alter the n-grams of this second melody, almost always dropping the similarity value to below 1. Given that most of the n-grams stay the same, the similarity value will not drop a lot, and thus it can deal with these alterations well.

A perk of this algorithm is that the ‘IDF’ calculation calculates the perceived ‘rarity’ of an n-gram by measuring how often other songs generally contain this n-gram. Having a rare n-gram in common between both melodies generally gives a higher similarity value than having a common n-gram in common.

The algorithm does not keep in mind the length of the sample, nor rhythm.

The Comparative Method This is an algorithm that incorporates rhythm and has been proposed by (Pinter, 2015). Here, we divide the melody into sets of note-rhythm tuples. The notes are represented as pitches manually transposed to the same key, and the rhythmic aspect is the starting time of the note represented in the amount of quarter notes that have passed from the start of the sample until the start of this note (for the first note always 0).

For all the note-rhythm tuples of A, we test whether the tuple is also found identically in B. The fraction of tuples in the original melody A that match perfectly with a tuple in melody B is considered the similarity value.

Formally defined, this is the formula:

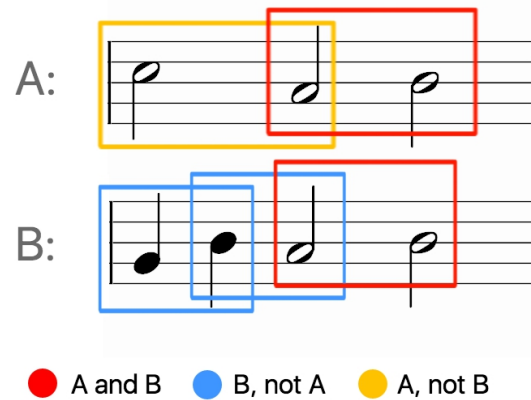


Figure 2: The Tversky.plaintiff.only algorithm is run with the top melody (A) and bottom melody (B). We suppose we look at 2-grams here, and thus $n = 2$. Melody A contains two 2-grams. The first 2-gram is yellow because it only exists in A and not in B. For similar reasons the first two 2-grams of B are blue, since they are not found in melody A. The last 2-gram of both A and B is red, because it exists in both A and B. We assume that the yellow 2-gram is very common and found in 70% of all melodies in an arbitrary database C, containing 100 melodies. The red 2-gram is less common and only occurs in 50% of the songs in C. Now, $\frac{IDF_C(RED)}{IDF_C(RED)+IDF_C(YELLOW)} = \frac{\log(\frac{100}{50})}{\log(\frac{100}{50})+\log(\frac{100}{70})} = \frac{0.30}{0.30+0.15} = 0.33$.

$$\frac{\sum_{(a,b) \in K} \begin{cases} 1 & \text{if } (a,b) \in B \\ 0 & \text{else} \end{cases}}{|K|}$$

where

$$K = \{(a,b) | (a,b) \in A\}$$

and where a is the transposed pitch (from 0-11), b is the amount of quarter note passed from the start of the song until the start of this note, A is the first melody, and B is the second melody.

An example of The Comparative Method at work is shown in Figure 2.

Evaluation Additions do not change the output if the addition does not mess up the rhythm of the existing notes. Deletions do change the output, but only slightly, identically to substitutions. The algorithm however is very prone to slight changes in pitch or rhythm, which the previously mentioned algorithms are not. As such, though incorporating rhythm, it cannot deal well with slight rhythmic variations. It also does not keep in mind the commonness of note

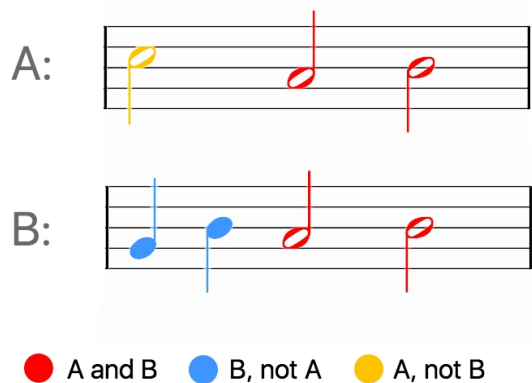


Figure 3: The Comparative Method is run with the top melody (A) and bottom melody (B). In tuples, A would be ((3, 0), (0, 2), (2, 4)), while B would be ((10, 0), (2, 1), (0, 2), (2, 4)). The first tuple of A cannot be found in B (though there is a note that matches solely in rhythm), and the second and third tuple of A can be found identically in B. The eventual similarity value is $\frac{2}{3}$ or .66.

progressions, nor the length of the sample.

We should note that the examples shown are merely to aid understanding but do not give the same outputs as the implementation we will use. In the version of the algorithm we will be using, we are not looking at notes directly, but rather at the intervals between them. Examples of what might change using our implementation is given in the 'Similarity Algorithms' section under 'Intervals'.

Methods

The Dataset

The data consists of 1,099 pop songs, retrieved from the pop section of the MidiWorld website³, and 58 remixes of the song 'Megalovania' by Toby Fox, obtained from the MuseScore website⁴. Though Megalovania is not a pop-song, it has a clear and repeated melody, and was one of few songs with many remixes available.

All songs are obtained in MIDI format. Midi is a music format that does not store sounds, but rather 'event data'. This event data can include note pitches, note velocities (loudness), instruments, etc., similarly to what would be written on sheet music. This data can then be translated to sounds when read by a program. This makes it easy to change and read songs. Audio or video files do store sounds, which often results in noisy files which change a lot with every performance

³<https://www.midiworld.com>

⁴<https://musescore.org>

of the song. It also requires separation of the instruments to get the melody line. Every of these problems are taken care of when using MIDI files.

Representation

The pre-processing steps have been discussed in the previous paragraph. We will quickly discuss the implementation of them.

In case there are chords, the top note of the chord will be used for the melody. At any point, we only look at one track (or: one individual instrument) of the song, and getting rid of chords ensures that we are dealing with only one note at a time. We ignore all rhythmic aspects, including duration of notes, rests, and repeated notes. This is done by using the Voice Normalization (Vnorm) algorithm suggested by (Rigaux, 2019). Lastly, we will take the intervals between the notes. This gives us a representation of interval-sequences.

We do not know what track contains the melody, and finding the track that contains it can be difficult and inconsistent. For this reason, we run the algorithm for every track of a song and keep the best match, expecting only tracks with highly melodic material to be able to match the melody well.

Algorithm Implementation

The algorithm chosen for this experiment is the Edit Distance algorithm. It is an intuitive, simple and widely applicable algorithm which scores well on many of the requirements.

The chosen operations are the Levenshtein distance operations, including insertion, removal, and substitution.

The weights of each operation are 1 by default. For a general explanation of the Edit Distance algorithm, see the 'Algorithms' section.

Run and Evaluation

The Edit Distance algorithm is designed to match two strings. In our case, our original melody (melody A) is 10 characters long (8 characters in the final representation), while the melody it tries to match (melody B) can be a few 100 characters long, giving a similarity value of over 90. We assume that the best melodic match uses only a small portion of this melody. To find this best match, we adjust the starting and ending position of the algorithm. We do this as follows:

Firstly, we run the algorithm for every possible starting position in the melody. We pick the best match it finds for all these starting positions as our final match.

Secondly, since looping over all tracks and all starting positions can already be quite slow, we will not be looking at all possible ending positions too. The first step of optimizing the ending position is to, instead of looping over all ending positions, take the final note of the song as the ending position. From this, the algorithm will derive the optimal length during the computations.

To find this optimal length, it looks at the smallest amount of operations with at least one operation being applied to all characters of melody A. Since it calculates a matrix (for explanation of the algorithm, see the 'Algorithms' section), it

can simply look at the row which has the lowest value in the last column to get the optimal length. This lowest value is our similarity value. By looking at the last column we guarantee that at least one operation has been applied to all characters of melody A.

To summarize, if we take the final note of the song as the ending note of melody B, it will automatically derive the optimal ending point from this. However, this requires that we fill in the matrix from the current starting position until the end of the song, which can still be a few hundred notes in length. To fix this, we assume and expect the optimal length of melody B to be very much near the length of melody A, given that every added or deleted note is reflected by an extra point in the similarity value. This allows us to state the maximum length of the melody not as the final note of the song, but rather as equal to the length of melody A plus a fixed constant p , while assuming that almost no song will have its similarity value surpass a value of p . In practice, we made p equal to the length of melody A, thus resulting in $p = 8$. As can be seen in Figure 5, no song had a similarity value higher than 7, justifying our choice.

For an example of all these steps, see Figure 4.

The algorithm returns a similarity value for every track of a song. The highest similarity value of all tracks is used as the similarity value of that song. All of these values are stored and then represented in a graph, showing the remixes vs the non-remixes. From here, both of the groups are fit with a Gaussian function, and an optimal separation boundary is found. This boundary represents the difference between plagiarized songs and non-plagiarized songs. This method has been suggested by (Casey, 2007).

Results

The results of the experiment are shown in Figure 5. The optimal separation boundary between the remix and non-remix classes can be found at a similarity value of around 2.0. This suggests that melodies with a similarity value of 2 or lower will be plausibly considered plagiarism, and melodies with a similarity value of 3 or higher are less plausibly considered plagiarism.

As an example of this boundary, Figure 6 compares two melodies retrieved in the experiment. The top melody is the original (melody A), the middle is a remix with a similarity value of 2 (melody B) and the bottom ones show two variants of an ordinary song with a similarity value of 3 (melody C). Some things are immediately clear when looking at this example. The rhythmic aspect we chose to ignore clearly creates a significant difference in the way the two variants of melody C look, to the point that we would likely not have spotted the similarities ourselves, not even after listening to the pieces. Recomposing the melody to fit the rhythm of the original melody makes the similarity clear in both looks and sound.

The choice of ignoring rhythm is put to the test here, given

		0	1	3
	0	1	2	3
0	1	0	1	2
1	2	1	0	1
2	3	2	1	1
3	4	3	2	1
4	5	4	3	2
5	6	5	4	3
6				
7				
8				

Figure 4: The Edit Distance algorithm as implemented. We convert melody A (0,1,3) (x-axis) into melody B (0,1,2,3,4,5,6,7,8) (y-axis). The most top-left 0 is the starting point, representing (0,1,3). Moving right represents deleting a character of melody A, moving down represents adding a character of melody B, and moving diagonally to the bottom-right represents substituting a character from melody A for one of melody B. The values represent the minimal amount of operations in order to get to that state. Melody B is capped at twice the length of the first melody, with $p = 3$. The minimum value obtained in the last column is the similarity value, which is 1. This makes lengths of 2, 3 and 4 the optimal lengths of the melody.

that the song does not sound similar on a first glance. If we cannot hear any resemblance, the algorithm is clearly not working as intended. However, if we match the rhythm, the similarities become clear, suggesting that rhythm is more important than anticipated.

Sadly, because only 58 remixes were found, the accuracy and precision of the experiment are not significant and therefore not presented.

Conclusion

We have discussed the complications of choosing an algorithm to aid in finding plagiarism by providing objective evidence of substantial similarity. Several requirements that such an algorithm needs to meet have been discussed, including the commonness of a melody, the possibility of adding, subtracting or altering notes, and whether rhythm is important. We have selected some common similarity algorithms

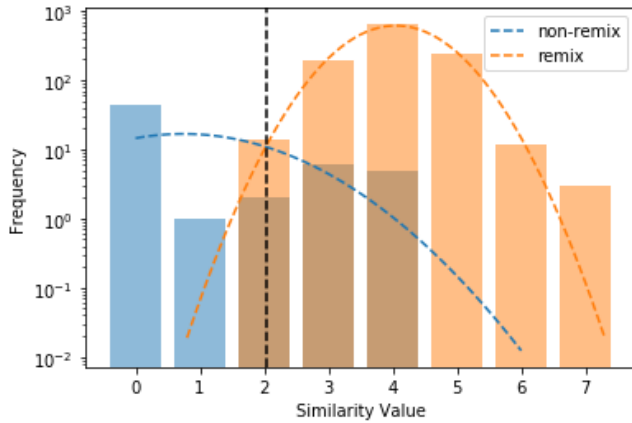


Figure 5: The distribution of similarity values of remixes and non-remixes, as compared to the song ‘Megalovania’ using the Edit Distance algorithm. The Gaussian Models fit the distributions and show an optimal threshold between the two classes at around value 2.0. Though unintuitive, the lower the similarity value, the more similar two songs are considered.

and analyzed whether they meet the requirements we proposed. The Edit Distance algorithm has been chosen to be tested on both remixes and non-remixes. The optimal boundary between these is considered the optimal threshold for plagiarism, and this threshold lies around the 2.0, suggesting that songs with a similarity value of 2 or less are plausibly plagiarized, and 3 or higher are implausibly plagiarized.

Given the low amount of remix data we were able to find, no strong conclusions can be drawn from this, and measuring the accuracy and precision would not be meaningful. This was an explorative study and we have laid down some important factors that need to be considered when searching for algorithms that can determine melodic plagiarism. Using a larger database of MIDI remixes to test the algorithms on is required to give more meaning to the results.

Rhythm has been ignored for a multitude of reasons, including time-constraints, a lack of existing algorithms incorporating it, and contradicting thoughts on its importance. However, the example of the song ‘Any Other Way’ showed the significance of including rhythm in defining similarity. Only when matching the rhythmic aspects, did the similarity become clear. Rhythm seems to be important enough to spend future research on. Other aspects like harmony, structure, and functional harmony, attempting to aid the algorithm, can also be explored. This paper only attempted to measure how well current similarity algorithms perform on this task, but given the many factors involved in deciding plagiarism, we could see new algorithms be designed for the purpose of determining plagiarism using the requirements we proposed, including the aspects named above. Examples of added features include: a rhythmic feature as suggested by (Rigaux, 2019), where all melodies with identical similarity value will be ranked according to their rhythmic similarity to the origi-



Figure 6: The original melody, ‘Megalovania’, is shown on the top (A). In the middle is shown a remix, conveniently titled ‘Megalovania remix’ (B), with a similarity value of 2 (plausible plagiarism), and on the bottom is shown a piece of Celine Dion’s ‘Any Other Way’ (C), which has a similarity value of 3 (implausible plagiarism). Two variants are shown of this last melody - the original and an altered version to match the rhythm of the original. The red notes are the notes which follow a wrong interval.

nal melody; comparing (parts of) the melody to a database of melodies to find its commonness, as derived from the IDF algorithm; multiplying the output of an algorithm by a sigmoid function applied to the length of the melody, to incorporate the length of the melody into the similarity value.

The method of comparing remixes to non-remixes is useful for viewing the potential of the algorithm and simultaneously for defining a threshold, yet there are more methods which can be considered. Examples of such methods include a 1-to-1 comparison of an algorithm’s computational steps with steps performed in actual musical similarity analyses done by experts, or merely comparing the output of algorithms with the output of court cases, as done by (Müllensiefen, 2009).

As it stands currently, no similarity algorithm seems ready for the task of defining plagiarism. No algorithm considers all of the factors included in defining plagiarism yet, though some algorithms do show promising results, and the requirements for them have become clear. To that end, current similarity algorithms are not yet applicable in defining a threshold for plagiarism cases.

References

Beaumont-Thomas, B. (2020, March 18). *Katy perry wins appeal in \$2.8m plagiarism case*. Retrieved from <https://www.theguardian.com/music/2020/mar/18/katy-perry-wins-appeal-in-28m-plagiarism-case-dark-horse>

Casey, M., Michael; Slaney. (2007). Fast recognition of remixed music audio. In *Acoustics, speech, and signal pro-*

- cessing (pp. 1025–1028). Hanover.
- de León; Antonio Pertusa; Carlos Pérez-Sancho; José Manuel Iñesta Quereda, D. R. P. J. P. (2006). Melody track identification in music symbolic files. In *Flairs conference*.
- Dingfelder, S. F. (2006, June). Name that tune. *Monitor on Psychology*, 37(7), 52.
- Foscarin, F. F.-S.-R., Francesco; Jacquemard. (2019). A diff procedure for music score files. In *6th international conference on digital libraries for musicology* (p. 58–64). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/3358664.3358671> doi: 10.1145/3358664.3358671
- Frieler, F., Klaus; Riedemann. (2009). Is independent creation likely to happen in pop music? In *Proceedings of the 7th triennial conference of the european society for the cognitive sciences of music* (pp. 141–146). Jyväskylä.
- Giere, C. d. (z.d.). *Wicked's musical themes*. Retrieved from <http://www.musicalschwartz.com/wicked-musical-themes.htm>
- Majumder, B. D., Sanjay; Smith. (2018). Real time pattern based melodic query for music continuation system. In *Audio mostly 2018: Sound in immersion and emotion* (pp. 1–5). Wrexham: Association for Computing Machinery.
- Müllensiefen, M., Daniel; Pendsch. (2009). Court decisions on music plagiarism and the predictive value of similarity algorithms. In *Proceedings of the 7th triennial conference of the european society of the cognitive sciences of music* (pp. 207–238). Jyväskylä: Musicae Scientiae.
- Neely, D. R.-N., Adam; Riehl. (2020, February 10). *Every melody has been copyrighted (and they're all on this hard drive)*. Retrieved from https://www.youtube.com/watch?v=sfXn_ecH5Rw
- Norman, H. (2014). *Intellectual property law* (Second ed.). Oxford, Oxfordshire, United Kingdom: Oxford University Press.
- Pettersson, E. (2019, September 23). Music copyright law in search of new standard for infringement. *Insurance Journal*.
- Pinter, D. (2015, September). *Plagiarism or inspiration?* Retrieved from http://www.icce.rug.nl/~soundscapes/VOLUME18/Plagiarism_or_inspiration.shtml
- Rigaux, N., Philippe; Travers. (2019). Scalable searching and ranking for melodic pattern queries. In *20th international society for music information retrieval conference* (pp. 801–808). Delft.
- Stav, I. (2014). Musical plagiarism: A true challenge for the copyright law. *DePaul Journal of Art, Technology Intellectual Property Law*, 25(1), 1–52.
- Uitdenbogerd, J., Alexandra; Zobel. (2002). Manipulation of music for melody matching. In *Multimedia '98: Proceedings of the sixth acm international conference on multimedia* (pp. 235–240). Melbourne: Association for Computing Machinery.
- Ukkonen, K. L. E. (2000). Including interval encoding into edit distance based music comparison and retrieval. In *Proceedings of the symposium on creative and cultural aspects and applications of ai and cognitive science (aisb'00)* (pp. 53–60).