Application of Smith-Waterman to detect musical plagiarism

Benjamin Dizdar

Jeevan Sivamohan

584A Algorithms for Bio-sequences

May 2nd, 2025

1. Abstract

In copyright law and music industry litigation, determining whether one piece of music plagiarizes another remains largely subjective and manual. Our project addresses this problem by developing a computational framework for detecting melodic plagiarism using string alignment techniques adapted from bioinformatics. Specifically, we represent melodies as sequences of pitch intervals and rhythmic patterns and apply modified local alignment algorithms, such as Smith-Waterman and Dynamic Time Warping (DTW), to detect shared motifs while accounting for musical invariants such as transposition, tempo variation, and repetition.

As a case study, we examined the widely recognized melodic similarity between Puff Daddy's *"I'll Be Missing You"* and Sting's *"Every Breath You Take."* Although both songs clearly share a 14-note countermelodic motif, our Smith-Waterman implementation failed to recover the match due to its rigidity in handling temporal offsets, yielding a maximum score of only 2. In contrast, DTW successfully aligned the motif by allowing flexible, non-linear warping of the time axis, illustrating the need for more adaptive alignment models in musical contexts.

Our findings underscore that while pitch-interval encoding and symbolic preprocessing improve robustness and interpretability, traditional alignment models like Smith-Waterman remain limited in musical settings where motif embedding and timing shifts are common. These limitations reflect broader challenges in Music Information Retrieval (MIR). As highlighted by Typke et al. (2005), most existing systems struggle to detect musically meaningful transformations and are constrained to either overly general or overly specific retrieval tasks. Melody-based plagiarism detection remains a difficult, unresolved computational problem.

Nonetheless, our implementation achieved notable performance: a single track could be queried against a 40-million-event database in roughly 25 minutes using our parallelized C++ backend. This efficiency lays a strong foundation for future extensions that incorporate DTW or hybrid scoring models to improve precision in identifying temporally displaced musical reuse.

2. Problem Description and Motivation

Detecting musical plagiarism remains a significant challenge in computational musicology, copyright enforcement, and music information retrieval. While music often borrows from shared conventions, distinguishing between inspiration and infringement is subjective and typically requires expert testimony or manual comparison. Our goal is to develop an automated system that can detect melodic plagiarism through approximate symbolic alignment. Inspired by tools in bioinformatics, we treat melodies as structured sequences of pitch intervals and rhythmic durations.

Musical plagiarism is especially difficult to detect because plagiarized melodies rarely copy notes exactly. Instead, they often preserve abstract structural features such as ascending or descending intervals, common rhythmic figures, or chord progressions. Our system addresses this

challenge by using pitch interval encoding, motif-based compression, and local sequence alignment to identify melodic similarity in a transposition-invariant way.

From a stringology perspective, our problem is to efficiently compare symbolic sequences that represent melodies. The system must handle approximate local alignment, transpositional variance, and rhythmic flexibility while scaling large datasets. Our implementation also supports polyphonic MIDI data by isolating candidates for melody-carrying lines and distinctive tracks using a filtering pipeline and aligning compressed motifs to detect potential borrowing.

3. Literature Review

One of the most influential applications of string-matching algorithms to symbolic music was introduced by Mongeau and Sankoff (1990). They proposed a generalized edit distance model that evaluates melodic similarity through dynamic programming, representing each monophonic melody as a sequence of relative pitches and normalized durations. Their model incorporates perceptual heuristics, assigning lower penalties for consonant substitutions like octaves and fifths, and higher penalties for dissonant intervals, making the algorithm robust to common musical transformations such as transposition or tempo variation. Later, they extended this approach to a local alignment framework like Smith-Waterman, capable of identifying highly similar subregions within otherwise dissimilar melodic sequences.

In their 2008 paper, W. Bas de Haas and colleagues introduce the Tonal Pitch Step Distance (TPSD), a novel similarity measure for chord progressions based on Lerdahl's Tonal Pitch Space (TPS) a cognitively grounded model of harmonic distance. The authors propose a method for representing chord sequences as step functions that track each chord's harmonic distance from the tonic triad. The distance between two chord sequences is then defined as the minimal area between their step functions, normalized for length and aligned using cyclic shifts. This approach allows for key-invariant, partial matching and yields results that align well with human intuition, particularly in the analysis of jazz standards and blues variations. Their experimental results on 388 jazz progressions from the *Real Book* demonstrate the TPSD's effectiveness in retrieving harmonically similar variants of songs, despite structural or harmonic substitutions.

Other popular computational strategies for detecting melodic similarity and plagiarism. Nguyen et al. (2023) categorize these into three major groups: string-matching methods (e.g., edit distance), statistical sequence modeling using N-grams and TF-IDF-based measures, and supervised machine learning algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM). Edit distance, for instance, measures the number of operations needed to transform one melodic string into another but often struggles with large-scale musical transformations like key modulation or phrase reordering. To address this, a variant using bipartite graph matching has been proposed, where melodic fragments are treated as graph nodes and similarity is computed through maximum-weight matching—incorporating music-theoretic parameters such as pitch, duration, and consonance. Meanwhile, N-gram models, commonly used in textual information retrieval, have been adapted to encode sequences of pitch intervals or

symbolic note events, with similarity measured via frequency overlaps (e.g., Sum Common, Ukkonen distance, Tversky index).

4. Implementation

In constructing a reliable pipeline for detecting musical plagiarism from symbolic MIDI data, it is essential to first isolate the tracks, most likely to contain melodically relevant material. Melodies are typically the source of contention for music plagiarism as they are typically the most well recognized and core part of any given song. We implemented a set of heuristic filters both for computational efficiency and to highlight tracks which were more likely than others to have definite aspects of plagiarism. These filters aim to exclude accompaniment and percussion tracks that contribute minimally to melodic identity, allowing us to focus alignment methods (such as Smith-Waterman) on tracks with the greatest potential for motif reuse.

Our first principle is that melody-carrying lines tend to exhibit a wide range of pitch movement and rhythmic variation, in contrast to percussive or supporting harmonic parts. Melodies often display a balance of pitch diversity, phrase structure, and durational contrast, distinguishing them from bass lines, rhythm loops, and harmonic arpeggios.

To formalize this, we applied three key filters:

Average Note Duration Threshold: Tracks with extremely short note durations (e.g., ≤ 72.5 ms) are characteristic of percussive or quantized rhythmic content, rather than expressive melodic lines. These often stem from MIDI drum mappings or step-sequenced instruments, which play a supportive rather than thematic role.

Pitch Monotony: Tracks in which over 80% of notes are a single pitch likely correspond to pedal tones, rhythmic accents, or drum hits mapped to a single MIDI note. Such elements, while musically important in texture or groove, contain little melodic information and do not typically constitute motifs relevant to plagiarism detection.

Local Pitch Diversity (Sliding Window): We slide a window over each pitch sequence and measure the number of unique notes per segment. If more than 80% of these windows contain fewer than three unique pitches, the track is flagged as repetitive or ostinato-based pattern often found in bass lines/ These parts usually do not express melodic ideas independently of the main line.

Additionally, we implemented a is_monotonous_track check that filters out tracks with highly regular note spacing and duration, indicative of metronomic loops or drum patterns. These tracks typically lack the expressive timing fluctuations and durational variation seen in melodic phrasing.

Empirically, these filters removed approximately 40% of tracks across our dataset, significantly reducing the volume of non-melodic content without discarding musically meaningful material. We worked under the assumption that typically the melody is within a small

subset of instrument tracks (often just one or two) that carry the core melodic identity of the piece. However, we acknowledge a key limitation of this approach: the melody may alternate between multiple instrumental or vocal lines (e.g., call-and-response between voice and synth, or switching between verse and chorus leads). Detecting and tracking these interleaved melodic voices presents a non-trivial computational challenge, especially when instrumentation or timbre is not consistent across sections.

Current filtering heuristics are not designed to model such temporal or structural dependencies between tracks, and as such, melodic fragmentation across voices may lead to partial exclusions. While our goal was to retain the most relevant and continuous melodic line(s), solving this problem in full would likely require more advanced techniques such as melody tracking with inter-voice continuity constraints, or unsupervised clustering methods applied to symbolic features across tracks.

Identifying the melody within polyphonic symbolic music data remains a non-trivial and open-ended problem in music information retrieval. Unlike tasks such as key or tempo detection, which are relatively well-defined and often globally consistent within a piece, melody identification involves discerning the most outgoing voice in a multi-instrumental texture, a task complicated by the fluidity and subjectivity of what constitutes a "melody." The melodic role can shift dynamically across sections (e.g., vocals in the verse, lead guitar in the chorus), alternate between voices (call-and-response, layered counterpoint), or even temporarily disappear in favor of rhythmic or harmonic content.

Symbolic representations like MIDI offer detailed note-level data (pitch, onset, duration), but are often unordered or ambiguously labeled, and may contain multiple voices interleaved within the same channel or spread across several tracks due to differences in instrumentation. These structural ambiguities make it difficult to algorithmically determine which voice carries the melody at any given point in time.

Even when heuristics like pitch range, rhythmic variability, or note density are used, these cues can be misleading in genres where the melody is rhythmically flat (e.g., blues) or narrow in range (e.g., minimalist electronic music). Our own filtering system, though informed by empirical patterns, occasionally misclassified melody-carrying lines, particularly in stylistic edge cases such as monotonic rock vocals (e.g., *ZZ Top*) or instances where the melodic theme is embedded within repetitive patterns.

In our early experiments, we explored the use of chroma encoding to represent symbolic pitch data, motivated by its desirable property of octave invariance. Chroma features reduce each pitch to its corresponding class within the 12-tone chromatic scale (i.e., C, C♯/D♭, ..., B), effectively collapsing pitches separated by one or more octaves into a common representation. This is advantageous in many musical tasks, such as chord recognition, where harmonic identity is preserved despite transposition across registers. Chroma encoding also helps mitigate small

registration differences that may otherwise obscure similarity between two functionally identical phrases performed at different pitch levels.

However, we found that this approach introduced significant ambiguity when used as input to Smith-Waterman local alignment. Because chroma encoding reduces the pitch space to just 12 discrete values, it vastly increases the likelihood of spurious alignments. Two musically unrelated phrases with entirely different contours and structures may produce overlapping chroma sequences simply due to shared pitch classes. In our case, this resulted in many false positives, where the algorithm aligned phrases that were harmonically compatible but not melodically similar in any meaningful sense.

To address this, we transitioned to a pitch interval representation, which captures the relative movement between successive notes rather than their absolute or pitch-class identity. This preserves transposition invariance while retaining essential information about melodic contour and directionality, making it better suited for detecting structural reuse in symbolic melodies. Despite this approach being superior to the chroma encoded structure, there was still the issue of alignment without using DTW.

For example, consider the following pitch interval sequences extracted from two different tracks:

- Sequence A: [0, -2, -2, 4, -2, 2, -4, 0, **4, 1, -5, 0, 0, 5, -1, -2, 0, -2, 4, -4, 0, 4**]
- Sequence B: [1, -1, -2, -2, -5, 9, 1, -1, -2, -2, 0, 0, **4, 1, -5, 0, 0, 5, -1, -2, 0, -2, 4, -4, 0, 4**, -1, 3]

The highlighted segment is a 14-note motif that occurs identically in both sequences. However, in Sequence B, the motif is embedded with a shift, preceded and followed by unrelated material. When applying standard Smith-Waterman local alignment, the algorithm failed to recover this musically significant motif as part of the optimal alignment due to early score penalties from mismatches outside the motif region.

Admittedly, prefiltering and fitting with a technique like DTW, could have flexibly aligned sequences based on shape and contour, allowing it to successfully detect the matching motif even when surrounded by non-corresponding material. This example underscores that while pitch interval encoding enhances the robustness of our representation, it must be coupled with non-linear alignment techniques to realize its full potential in identifying melodic reuse.

Regarding the implementation and design choices of our Smith Waterman algorithm, the first step of the process was to build a system to navigate the directories and parse the json files for each track into a form that could be used to eventually run global and local alignment. After searching for a while we landed on the boost.filesystem library because it is compatible with the C++ 14 standard we were using and it has an easy interface. For the json parsing we use the nlohmann::json library because it had similar syntax to the python json parsing we had been doing,

and it made the conversion of the code a lot easier. Because we knew our goal was to a full search across the database, we ended up taking each track from the database and concatenating the pitch intervals together to form one long string comprising the whole database. This way we could do a single query against the whole database instead of having to do multiple queries across each song individually with its own initialization overhead and constant factors for maintenance, we could just do a single query over the entire concatenated database to save time. This introduces another problem of the possibility that the optimal alignment goes between songs in the database, to account for that between each song in the json parsing we put in the value -128 which is intended to represent a delimiter of sorts (like null terminator in C). The sigma function that is eventually used in the alignment algorithms gives negative infinity alignment in the case that the value being compared is -128. We checked to ensure no actual values in the json files were -128 so this should only be the case when it breaks tracks. This will ensure any alignments we find are solely confined to a single track.

Multiple approaches for global and local alignment were implemented and used for this project. First was a naive Needleman-Wunsch for global alignment. It used the large DP table and calculated alignment scores based on a given sigma function and gap penalty. However, because the goal of this project was to cross-compare tracks with an entire database worth of information, it was better to use a local alignment as compared to a global. We then implemented basic Smith-Waterman by taking the Needleman-Wunsch implementation and tracking the highest cell overall and backtracking until we reached a cell with value 0, indicating the end of the local alignment sequence. While the basic Smith-Waterman did what we needed it to do it was very space-intensive and not as quick as we would've wanted, especially over the whole roughly 10000 song databases.

We first sought to address the speed problem as best we could. We implemented several optimizations to the algorithm with the main ones converting the implementation from Python to C++ to get rid of the extra bounds checking and leverage C++ for its extra speed. We also used the existing C++ OpenMP library with compiler directives to parallelize the matrix calculations. Working along the anti-diagonal of the matrix, we were able to do multiple cell calculations at once. While we originally considered implementing the parallelization ourselves, we thought it was better to leave it to the OpenMP library as it has been used and tested rigorously and would be best for extracting performance from our system. In general, we saw about a 25% boost in performance from this. Our current theory is that we could have achieved a larger boost in performance if we were running it on a machine with more computing resources (threads, cores, etc). We also implemented other smaller changes like pre-allocating the vector to the necessary size to avoid the linear overhead of vector resizing and using move semantics and references wherever possible to avoid unnecessary copy overhead. While these smaller changes were individually not super impactful as a collective, they helped reduce the runtime of the program.

The next question was how we could limit the space. We started by reducing the size of the data representing the pitch intervals in the MIDI files. Originally, we had it that each interval was an int, which is native 4 bytes in C+,+ but represents way more data values than we needed. After

analyzing the alphabet, we figured out we could just use a char data type, only taking a single byte that still had sufficient data representation for our alphabet. This effectively reduced the necessary space by a factor of 4. We considered using the Hirschberg algorithm for linear space cost but eventually decided against it as we were taking some performance hits in terms of runtime because while it may be asymptotically the same runtime it has constant factor differences due to the calculation of the forward and backwards versions from the midpoint that made things a little bit too slow. It was also harder to implement and didn't parallelize as well as Smith-Waterman. We found that for reasonable track size comparisons we were able to keep the matrix in RAM, which we thought would be sufficient to maintain a good runtime.

We also analyzed the case in which the person may only want to know if it was plagiarism and what song it was plagiarized from instead of wanting the specific path of alignment for the plagiarism. As we know, if we only need the alignment score itself and not the path of alignment, we don't have to keep the whole matrix around and can instead keep only two rows (the row we are currently filling and the one directly above) as that is all that is necessary for the row update step of the algorithm, the only reason to keep the rest would be to do the backtracking for the alignment path at the end.

As such, Smith-Waterman was modified to only keep two rows in memory (condensing it down to linear space instead of quadratic) and filled the matrix while tracking which cell had the best alignment score and what that alignment score was. While this was good, we still needed a way to translate that cell position back into an actual track from which we maximally aligned. We therefore built an additional map into json parsing that maps the interval of the massive, concatenated char vector with the track it represents. This means that when we find a specific cell, the x value of the cell will correspond with the column representing the song we are aligning with (remember that because of the delimiter between songs we can only maximally align with a single song), we can just query the map to find which interval this belongs to. The query itself is based on an interval class we made with a set start and finish index. Because we know each of the intervals are disjoint, we can simply create a new interval with start and finish being the singular x value of the best alignment and the less than operator we defined for the class will only match on the interval that x value is defined within in the map. We used a binary search tree map implementation as opposed to hash map implementation because we needed to be able to binary search on the interval values to make the find time feasible (because we weren't searching by the specific key itself, just a single index we have no way to find what the actual start and stop times of the interval are so we can't do basic hashing). We also used a couple more small optimizations to reduce the copy overhead of the two-row approach to Smith-Waterman like alternating which row was the current row and which row was the previous row instead of copying one into the other to reduce copy overhead.

5. Results

After finding a feasible case study of plagiarism between Puff Daddy's *"I'll Be Missing You"* and Sting's *"Every Breath You Take"*, we manually extracted the two tracks that were most contentious—those carrying the main melody as well as the recognizable countermelody embedded within the instrumental. The 14-note motif discussed earlier emerged from these countermelody comparisons. While this motif was clearly present in both pieces, our implementation of Smith-Waterman was ill-suited to detect it effectively due to the temporal nature of the sequences. Specifically, Smith-Waterman assumes that similar elements occur at relatively consistent positions in both sequences and penalize early mismatches heavily, causing them to

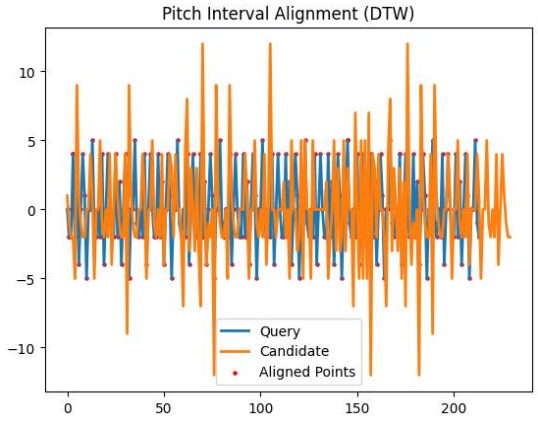

Figure 2: DTW Implementation between Puff Daddy's "I'll be Missing You" Track 1 Faith1, and Sting's "I'll be Watching You"- Track 2. DTW Distance: 479.0, DTW Path Length: 273, with Mean Distance 1.75 indicating high similarity

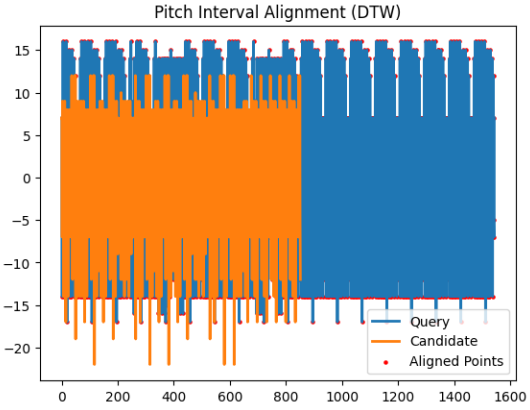

Figure 1b: DTW Implementation between Puff Daddy's "I'll be Missing You" Track 2 Guitar and Sting's "I'll be Watching You" Track 1. DTW Distance: 12530.0, DTW Path Length: 1545. Indicating less similarity, however, this is also a weakness of DTW's ability to fully register and adapt to the transposition of the music material.

miss alignments when motifs are temporally offset or embedded within non-matching material. As a result, the maximum Smith-Waterman score we obtained was only 2, far below what would be expected for a musically significant match. To address this, we explored the use of Dynamic Time Warping (DTW), which allows for flexible, non-linear alignments by warping the time axis to match similar contour shapes even when they occur at different rates or positions. In this case, DTW successfully aligned the shared motif, confirming its musical significance and illustrating that DTW would have been a valuable addition to our alignment pipeline for capturing temporally shifted motifs that Smith-Waterman alone fails to recover.

This limitation highlights a broader challenge: evaluating the effectiveness of our plagiarism detection system is difficult without a reliable set of ground-truth alignments. Smith-Waterman, while efficient and interpretable, is inherently rigid in its treatment of sequence timing and often fails to register perceptually meaningful matches that are structurally offset. As such, although the algorithm provided a solid foundation for scalable local alignment, it often underrepresents musical similarity where timing variation plays a key role.

This is not a problem unique to our approach. A broader review of symbolic music retrieval systems by Typke et al. (2005) reveals that despite a diversity of methods, ranging from string and set-based matching to probabilistic modeling, many MIR systems remain limited in their capacity to robustly detect transformations such as transposition, rhythmic variation, or motif embedding. They emphasize the need for a standardized evaluation framework (such as MIREX) and highlight that most systems are designed either for very specific or very generic tasks, leaving a gap when it comes to nuanced intertextual tasks like plagiarism detection. This further underscores that melody-based plagiarism remains an inherently difficult computational problem, with no single solution proving universally effective across genres or representations.

Nonetheless, from a performance standpoint, our implementation proved to be highly efficient. We were able to query a single track against the entire dataset, approximately 40 million events in total length, within about 25 minutes on a multi-core machine using our optimized C++ backend with OpenMP parallelism. This level of scalability is notable given the size and complexity of the symbolic input. However, without a more expressive alignment model like DTW incorporated directly into the main pipeline, our system's precision in identifying musically relevant reuse remained limited.

Future Implementations and Improvements

Inspired by TPSD and local alignment approaches, we initially sought to extract full chord progressions from MIDI files to compare harmonic structure using a Smith-Waterman alignment. However, implementing reliable chord extraction proved difficult. Our custom-built methods, as well as external tools such as music21, frequently failed to identify consistent or meaningful chords across different tracks, often producing ambiguous, overly complex, or unrecognized chord symbols. This inconsistency made it difficult to apply any alignment algorithm that assumed a standard symbolic representation. Furthermore, attempts to encode these extracted chord sequences into alignment-friendly formats were hindered by the unstructured nature of real-world MIDI data, which lacks consistent key annotation and often contains simultaneous polyphonic events that defy straightforward reduction to symbolic chord labels.

Beyond technical obstacles, we also recognized that melodic similarity cannot always be meaningfully assessed in isolation from its harmonic context. A melody that is harmonized differently, placed over a new chord progression, may share pitch-level features with an original but feel fundamentally different to listeners. This is because chord progressions provide a tonal framework that shapes how a melody is perceived. Even if two melodies contain the same intervals or contour, their alignment over different harmonic backdrops can drastically alter the emotional and stylistic impression they create.

This introduces a layer of subjectivity to our task: musical similarity is not solely a function of shared symbolic content, but also of how that content is interpreted within a broader musical context. What might appear plagiarized in a note-by-note comparison may not sound similar to listeners if the harmonic setting diverges significantly. Conversely, subtle changes in pitch or

rhythm may still evoke a strong sense of familiarity if the underlying chord structure remains intact. Because our current approach focuses on melodic surface features without explicitly modeling harmonic support, we acknowledge this as a limitation. Incorporating harmonic alignment in future work, possibly through a custom penalty system or more robust chord recognition models, could enhance the musical relevance and perceptual validity of our similarity assessments.

To improve the implementation of Smith Waterman with these challenges in mind, it is essential to tailor the algorithm's scoring matrix to reflect both music-theoretic structure and symbolic data uncertainty. While our current implementation evaluates melodic similarity based solely on pitch intervals, integrating harmonic information could help resolve ambiguities in motif comparison by capturing how melodies function within their tonal environment. One promising direction would be to build a domain-aware substitution matrix that leverages known harmonic relationships—such as shared pitch classes, proximity in the circle of fifths, or functional similarity (e.g., treating ii–V–I substitutions as low-cost mismatches). Unknown or malformed chords (e.g., "N.C." or symbols not recognized by extraction tools) could be soft penalized with a mild gap or mismatch score rather than a hard zero. Another useful extension would involve partial chord matching, where alignments between chords with overlapping pitch classes (e.g., Cmaj7 and C7) receive partial credit. Additionally, key-relative encoding could enhance transposition invariance by aligning chords based on their tonal role rather than absolute identity. These modifications would allow Smith-Waterman to become more robust in symbolic MIR contexts, bridging the gap between computational alignment and listener perception, and enabling it to better reflect the hierarchical and contextual nature of tonal music.

To make Smith-Waterman viable for chord progression data, the scoring matrix must be tailored to reflect both music-theoretic structure and data uncertainty. One promising direction would be to build a domain-aware substitution matrix that leverages known harmonic relationships—such as shared pitch classes, proximity in the circle of fifths, or functional similarity (e.g., treating ii–V–I substitutions as low-cost mismatches). Unknown or malformed chords (e.g., "N.C." or symbols not recognized by extraction tools) could be soft penalized with a mild gap or mismatch score rather than a hard zero. Another useful extension would involve integrating partial chord matching, where alignments between chords with overlapping pitch classes (e.g., Cmaj7 and C7) receive partial credit. Additionally, key-relative encoding could enhance transposition invariance by aligning chords based on their role within a detected or inferred key. These modifications would allow Smith-Waterman to be more adaptable in symbolic MIR contexts.

Bibliography

de Haas, W. B., Veltkamp, R. C., & Wiering, F. (2008). Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. In Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR) (pp. 51–56). http://ismir2008.ismir.net/papers/ISMIR2008_008.pdf

Mongeau, M., & Sankoff, D. (1990). Comparison of musical sequences. Computers and the Humanities, 24(3), 161–175. https://doi.org/10.1007/BF00053678

Nguyen, D., Vu, H., Do, L. M., & Hoang, K. (2023). A survey on melody similarity detection: Methods, challenges, and opportunities. Journal of New Music Research. (In press/preprint — update citation details when available)

Typke, R., Wiering, F., & Veltkamp, R. C. (2005). A survey of music information retrieval systems. In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR) (pp. 153–160). Queen Mary, University of London.