TIME-WARPED LONGEST COMMON SUBSEQUENCE ALGORITHM FOR MUSIC RETRIEVAL

AnYuan Guo *Hava Siegelmann*
University of Massachusetts, Amherst
Department of Computer Science

ABSTRACT

Recent advances in music information retrieval have enabled users to query a database by singing or humming into a microphone. The queries are often inaccurate versions of the original songs due to singing errors and errors introduced in the music transcription process. In this paper, we present the Time-Warped Longest Common Subsequence algorithm (T-WLCS), which deals with singing errors involving rhythmic distortions. The algorithm is employed on song retrieval tasks, where its performance is compared to the longest common subsequence algorithm.

1. INTRODUCTION

In recent years, a large amount of music has been made publicly available over the Internet. Various music collections come in formats such as MIDI, WAV, MP3, ABC, and GUIDO, to name a few popular ones. Furthermore, many programs exist that will convert between the different formats [15, 8, 1]. The proliferation of music data has driven up user demands for easy and efficient ways to search databases for a song of interest. A user can input a query via a keyboard or even by singing or humming into a microphone [6, 9, 13]. The user may have inaccurate pitch or rhythm, or may sing at a different speed than the original rendition. We introduce an algorithm that deals with rhythm and speed variations within the context of string-matching based music similarity metrics.

A piece of music is composed of a series of symbols. In this paper, we focus on monophonic music, in which at most one note is played at any given time. This class of music can be readily represented as a string over an alphabet, where the alphabet includes all the pitches that appeared within that piece of music. Given the representation of a piece of music as a string of symbols, the edit-distance based string matching algorithms, already widely employed in the speech and text processing communities [3, 18, 5], naturally lend themselves to the formation of plausible similarity measures.

One key strength common to string matching algorithms is that they are designed to deal with insertions, deletions and substitutions between the query string and the target, a property much needed considering the different types of errors that might be introduced in the retrieval process. There could be inaccuracies in the user's recall or inaccuracies in singing. Furthermore, errors are often introduced in the transcription process when the acoustic signals are turned into musical symbols [10]. Indeed, several string matching algorithms, such as approximate string matching, local alignment and the longest common subsequence algorithm have been successfully employed in music information retrieval systems [17, 13]. However, these algorithms do not have principled ways for handling speed variations and inaccuracies in rhythm. In this paper, we focus on extending the longest common subsequence algorithm with these additional properties.

Given two sequences, the longest common subsequence algorithm finds the longest subsequence the two have in common. The subsequence itself can be dispersed arbitrarily among each of the strings with gaps of non-matching symbols in between. Similarity measures based on this algorithm have been used to perform content-based music information retrieval [17]. It should be noted, however, that edit-distance based algorithms cannot deal with transposed matches [11].

According to studies in music recognition, the rhythm of a piece of music is important to recognition tasks [4]. The inclusion of rhythmic information has shown to improve music retrieval [17]. People often sing inaccurately with respect to rhythm - some notes are shrunk, others expanded. Furthermore, the rendition could be faster or slower than the original score. In other words, the duration of the notes sung do not match the original song, some nonlinear expansion/contractions could be introduced.
String matching techniques for song retrieval such as the longest commons subsequence algorithm lack a principled way to deal with the these variations [17] .

The contribution of this paper is the introduction of the time-warped longest common subsequence (T-WLCS) algorithm, which combines the desirable aspects of two algorithms, the dynamic time warping algorithm (DTW) and the longest common subsequence algorithm (LCS). The T-WLCS algorithm augments the LCS algorithm with the ability to account for songs played at different speeds, with possibly non-uniform expansions/contractions. At

the same time, it retains the desirable properties of the LCS algorithm, allowing for gaps between matching sequences. Finally, the algorithm is tested in a song retrieval application.

2. BACKGROUND

Dynamic time warping (DTW) is an algorithm developed by the speech recognition community to handle the matching of non-linearly expanded or contracted signals [14]. The algorithm finds the optimal path through a matrix of points representing possible time alignments between the signals. The optimal alignment can be efficiently calculated via dynamic programming.

Dynamic time warping operates as follows. Given two time sequences $X = \langle x_1, x_2, \ldots, x_m \rangle$, and $Y = \langle y_1, y_2, \ldots, y_n \rangle$, it fills an m by n matrix representing the distances of best possible partial path using a recursive formula:

$$D(i,j) = d(i,j) + min \begin{cases} D(i,j-1) \\ D(i-1,j) \\ D(i-1,j-1) \end{cases} \quad (1)$$

where $1 \leq i \leq m, 1 \leq j \leq n$, $d(i,j)$ represents the distance between x_i and y_j. $D(1,1)$ is initialized to $d(1,1)$. The alignment that results in the minimum distance between the two sequences has value $D(m,n)$.

Although DTW does have the flavor of the property that we desire, namely, it matches non-linearly stretched or compressed sequences, it is not directly applicable to the class of music retrieval tasks we are interested in. DTW aligns two sequences from beginning to end. Often, in music retrieval tasks, we are only given partial songs. In the next section, we will show how to augment the LCS algorithm with the time stretching properties of DTW.

3. TIME-WARPED LONGEST COMMON SUBSEQUENCE ALGORITHM

Due to inaccuracies and speed variations in singing, and the errors introduced in the music transcription phase, the query string and the stored song might differ. A desirable matching algorithm thus needs to take these factors into account.

String matching algorithms suitable for this task belong to the *edit-distance* family, defined as algorithms that find a minimum-weight sequence of edit operations (such as deletions, insertions and substitutions) that transform one string to the other. This family of algorithms is widely employed in real-life applications ranging from speech processing to molecular biology [7, 19].

3.1. The Original Longest Common Subsequence Algorithm

The longest common subsequence algorithm (LCS) belongs to this edit distance family of string matching algorithms. Specifically, the LCS algorithm finds the longest subsequence that two sequences have in common, regardless of the length and the number of intermittent non-matching symbols. For example, the sequences "abcdefg" and "axbydezzz" have a length four sequence "abde" as their longest common subsequence.

Formally, the LCS problem is defined as follows. Given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$, and a sequence $Y = \langle y_1, y_2, \ldots, y_n \rangle$, find a sequence Z, such that Z is the longest sequence that is both a subsequence of X, and a subsequence of Y. The subsequence is defined as a sequence $Z = \langle z_1, z_2, \ldots, z_k \rangle$, where there exists a strictly increasing sequence $\langle i_1, i_2, \ldots, i_k \rangle$ of indices of X such that for all $j = 1 \ldots k, x_{i_j} = z_j$ [2].

The solution to the LCS problem involves solving the following recurrence equation, where the cost for the edit operations is stored in c.

$$c(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c(i-1,j-1) + 1 & \text{if } i,j > 0 \text{ and } x_i = Y_j \\ max[c(i,j-1), c(i-1,j)] & \text{if } i,j > 0 \text{ and } x_i \neq Y_j \end{cases} \quad (2)$$

Using LCS as a similarity measure between two sequences has the advantage that the two sequences we are comparing can be of different length and have intermittent non-matches. In the context of music retrieval, this allows for the use of partial and noisy inputs.

3.2. The new algorithm

When a piece of music is expanded or compressed, we would like to recognize them as the same. For example, if 44556677 were matched against 4567, the output should be a high score. The LCS algorithm would output 4, since the sequences have the subsequence 4567 in common. Now, what if the sequences 42536172 and 4567 are given? The output of LCS is still 4, since the common subsequence is again 4567. But since the first pair of sequences are just expanded/contracted versions of each other, they should be considered as more similar than the second pair. The LCS algorithm can not make this distinction.

As we noted earlier, the dynamic time warping algorithm handles the expansion and contraction of the sequences but restricts the alignment to and end-to-end fashion, and is poor at handling insertions/deletions that might easily occur during music transcription. The LCS algorithm, on the other hand, handles extra/skipped characters and is very flexible in the starting/end points of the alignment, but does not deal with expansion/contraction. We take the desirable properties of each and construct the time-warped longest common subsequence algorithm (T-WLCS) that can deal with both sets of issues.

The recurrence formula for T-WLCS is:

$$c(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ max[c(i,j-1), c(i-1,j), c(i-1,j-1)] + 1 & \text{if } i,j > 0 \text{ and } x_i = Y_j \\ max[c(i,j-1), c(i-1,j)] & \text{if } i,j > 0 \text{ and } x_i \neq Y_j \end{cases} \quad (3)$$

Here $c(i,j)$ denotes the minimum cost *T-WLCS path* leading to the entry i,j in the *T-WLCS table*. The *T-WLCS table* is an n by m table that keeps track of the minimum-cost T-WLCS path leading to each possible alignment so far (see Figure 1 for an example LCS table and T-WLCS table). A T-WLCS path specifies an alignment between two strings. The *optimal T-WLCS path* is the one that achieves the minimum cost alignment. This cost is stored in the $c(m,n)$ entry in the table. The optimal paths for each of the examples are shown as highlighted squares in the table (see Figure 2).

3.3. Examples

Example 1 *S1 = "41516171", S2 = "4567". Compare the output of the two algorithms on the sequences S1 and S2. LCS(S1, S2) = 4, T-WLCS(S1, S2) = 4.*

For the example above, the two algorithms give identical results, since "4567" is the string they have in common. The T-WLCS table showing the run of the algorithm is shown in Figure 1(a).

Example 2 *S1 = "44556677", S2 = "4567". Compare the output of the two algorithm on the sequences S1 and S2. LCS(S1, S2) = 4, and T-WLCS(S1, S2) = 8.*

Here, the T-WLCS gives this matching a higher score than the pair in Example 1. This makes sense since this pair is actually more similar, because one is just a doubly stretched version of the other. The T-WLCS table is shown in Figure 1(b).

Example 3 *S1 = "4455661111177", S2 = "4567", compared the output of the two algorithms on sequences S1 and S2. LCS(S1, S2) = 4, T-WLCS(S1, S2) = 8.*

This shows that T-WLCS retains the advantage of the original LCS algorithm in that it can skip a section of text and match in a non-continuous fashion. The T-WLCS table for this example is shown in Figure 1(c).

4. EXPERIMENTS

We tested the effectiveness of our similarity measure calculated by the T-WLCS algorithm on a music information retrieval task. Songs from the Digital Tradition collection were used because this collection consists entirely

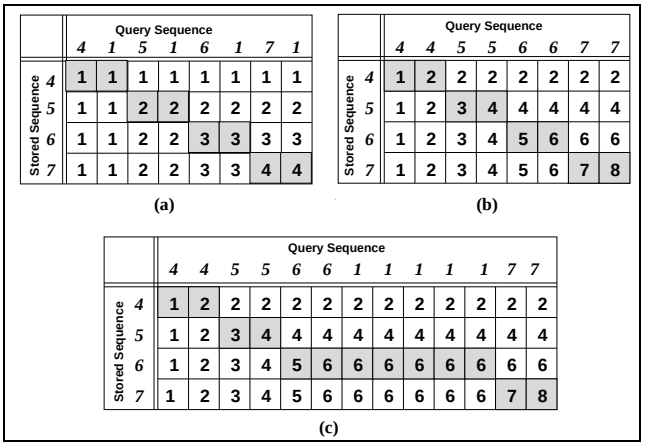

Figure 1. The T-WLCS tables containing $c(i,j)$ values for Examples 1, 2 and 3

of folk songs that are monophonic in nature. A subset of two hundred songs were randomly selected to be used in query/retrieval. For each song, we tested a range of possible query versions that simulate various types of errors. We stretched out the song by factors of up to four, also shortened and lengthened randomly selected notes to simulate inaccuracies in rhythm. Random insertions and deletion of notes were also made to simulate both singing errors and errors that are introduced during the music transcription process when acoustic signals are converted to musical notes. Different length queries were tested. Chunks ranging from 40 to 100 percent of the original song were randomly selected to use as query strings before the error simulations described above were used to perturb the songs.

The T-WLCS algorithm is used to calculate similarity scores between a given query sequence and each stored song in the database. The effectiveness of the retrieval system is measured by comparing the ranking of the original song (that the query is derived from) to all other songs in the database.

We compared the performances of the T-WLCS algorithm with the traditional LCS algorithm and found that the retrieval is more accurate when the T-WLCS algorithm is used to calculate the similarity score between two songs. See Figure 2 for details.

5. CONCLUSION AND FUTURE WORK

In this work, we presented the T-WLCS algorithm, which calculates similarity scores betweens songs, allowing for variations in speed and inaccuracies in the rhythm between the query and the stored music. The algorithm was employed on song retrieval tasks and compared favorably to the original LCS algorithm.

We plan to extend this work in several directions. First, we would like to fine tune the algorithm to include limits on the amount of allowable distortion between songs. On the application side, we would like to investigate the applicability of the T-WLCS algorithm to polyphonic music

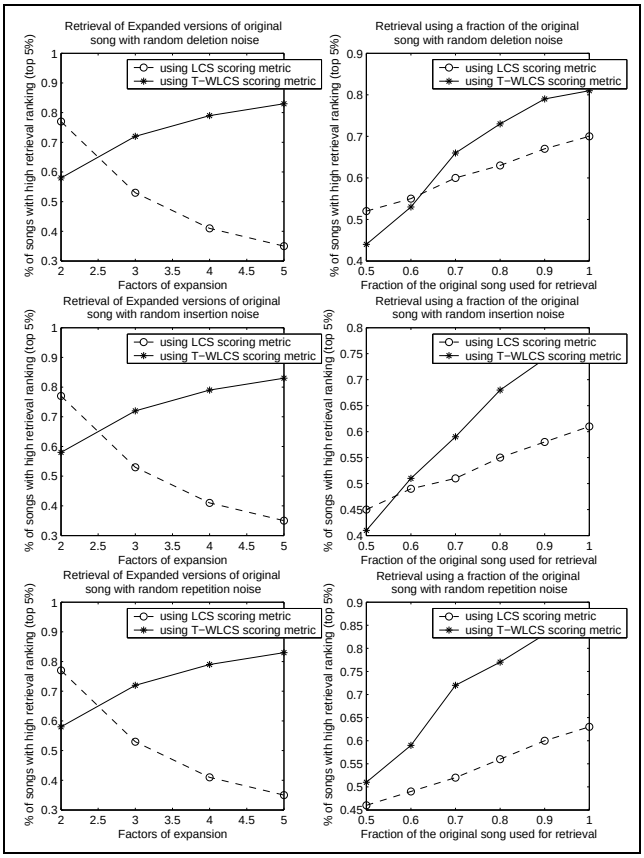

Figure 2. Experimental comparison between the LCS algorithm and the T-WLCS algorithm on song retrieval

retrieval. Since the algorithm inherently operates on linear sequences, we will experiment with methods that "flatten" out a polyphonic score to a linear theme [16, 12]. We also plan to apply the algorithm to clustering tasks that involve music recordings, for which rhythmic differences could also play an important part in the similarity measure.

6. REFERENCES

[1] Innovative music systems, inc. wav to midi, mp3 to midi converter - intelliscore. *http://www.intelliscore.net/*.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[3] F. Damerau. A technique for computer detection and correction of spelling errors. *Comm. of the ACM*, 7(3):171–176, 1964.

[4] D. Deutsch. Grouping mechanisms in music. In *Psychology of Music*, Orlando, 1982. Academic Press.

[5] J. French, A. Powell, and E. Schulman. Applications of approximate word matching in information retrieval. In *Proceedings of ACM CIKM'97*, pages 9–15, 1997.

[6] A. Ghias, J. Logan, D. Chamberlin, and B. Smith. Query by humming - musical information retrieval in an audio database. In *ACM Multimedia 95 - Electronic Proceedings*, 1995.

[7] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[8] K. Hamel. The salieri project - guido music notation. *http://www.salieri.org/guido/impl.html*.

[9] T. Kageyama, K. Mochizuki, and Y. Takashima. Melody retrieval with humming. In *ICMC Proceedings 1993*, 1993.

[10] A. Klapuri. Automatic transcription of music. In *Proceedings of Stockholm Music Acoustics Conference*, 2003.

[11] K. Lemstrom. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, 2000.

[12] A. Marsden. Modelling the perception of musical voices: a case study in rule-based systems. In *Computer Representations and Models in Music*, pages 239–263, London/San Diego, 1992. Academic Press.

[13] R. McNab, L. Smith, I. Witten, C. Henderson, and S. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the Digital Libraries Conference*, 1996.

[14] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-26(1):43–49, 1978.

[15] S. Shlien and B. Vreckem. The abc music project - abcmidi. *http://abc.sourceforge.net/abcMIDI/*, 2003.

[16] A. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In B. Smith and W. Effelsberg, editors, *Proceedings of the ACM Multimedia Conference*, pages 235–240, Bristol, UK, Sept. 1998.

[17] A. L. Uitdenbogerd and J. Zobel. Matching techniques for large music databases. In D. Bulterman, K. Jeffay, and H. J. Zhang, editors, *Proceedings of the ACM Multimedia Conference*, pages 57–66, Orlando, Florida, Nov. 1999.

[18] R. Wagner and M. Fisher. The string to string correction problem. *Journal of the ACM*, 21:168–178, 1974.

[19] T. Yap, O. Frieder, and R. Martino. *High performance computational methods for biological sequence analysis*. Kluwer Academic Publishers, 1996.