A Probabilistic Model of Melodic Similarity

Ning Hu, Roger B. Dannenberg, Ann L. Lewis

School of Computer Science, Carnegie Mellon University
email: ninghu@cs.cmu.edu

Abstract

Melodic similarity is an important concept for music databases, musicological studies, and interactive music systems. Dynamic programming is commonly used to compare melodies, often with a distance function based on pitch differences measured in semitones. This approach computes an "edit distance" as a measure of melodic dissimilarity. The problem can also be viewed in probabilistic terms: What is the probability that a melody is a "mutation" of another melody, given a table of mutation probabilities? We explain this approach and demonstrate how it can be used to search a database of melodies. Our experiments show that the probabilistic model performs better than a typical "edit distance" comparison.

1 Introduction

1.1 Sequence Matching

Sequence matching techniques are used in various research areas, including speech recognition, biological sequence analysis, and text information retrieval. Many algorithms and methods exist for such purposes, i.e. dynamic programming algorithms for approximate string matching, and algorithms associated with Markov Models (e.g. the Viterbi algorithm) and Hidden Markov Models (HMM).

As the dynamic programming technique is popular for approximate matching and alignment, it is only natural that it be broadly used in the area of music processing since melodic contours can be represented as sequences. Dynamic programming has been used for computer accompaniment (Dannenberg, 1984), comparison of melodic contour in improvisations (Stammen & Pennycook, 1993), the alignment of performances to scores for expressive timing analysis (Hoshishiba, Horiguchi, & Fujinaga, 1996; Large, 1993), and music search (McNab, *et al.*, 1996).

1.2 MIR and Query by Humming

The goal of Music Information Retrieval (MIR) is to process musical information and search music databases by content. One interesting branch of MIR is sometimes called "Query by Humming". Compared to other data entry methods, humming is considered the easiest way for ordinary non-musicians to express a music query. Unfortunately, one feature of sung queries is a high error rate exacerbated by a difficult transcription problem, so robust matching techniques are essential. We are interested in comparing and evaluating melodic matching techniques that compare sung queries to MIDI data. (In the future, we plan to investigate searching audio data as well.)

1.3 Probabilistic Model

To the best of our knowledge, previous applications of dynamic programming to music have relied upon an ad-hoc distance function. For example, a melodic comparison function might count the number of matches between the two melodies, or it might find minimum number of semitones of transposition required to make the melodies consistent.

The same basic algorithm can be viewed in probabilistic terms, as presented by Durbin, *et al.* (1998). In this approach, we estimate the probability that one melody is derived from the other. Viewing the melodies as sequences of symbols, we assume that each symbol of the derived melody is independently generated according to some joint probability p_{ab}, the probability that symbol b was derived from symbol a. Under this assumption, the probability of the entire sequence alignment is:

$$P(x,y) = \prod_i \frac{p_{x_i,y_i}}{q_{x_i} q_{y_i}}$$

Where p is the joint probability, and q is the probability that a symbol occurs independently. This ratio is known as the *odds ratio*. Because dynamic programming computes sums of distances, we take the logarithm of the odds ratio to get the *log-odds ratio*:

$$S = \sum_i s(x_i, y_i),$$

where

$$s(a,b) = \log\left(\frac{p_{ab}}{q_a q_b}\right)$$

In our work, we assume that any symbol is equally likely, so the q's become constants that we ignore. In the dynamic programming algorithm, the "edit distance" for substituting pitch a for pitch b

becomes -log(p_{ab}), where the minus sign is used simply so we can view this as a "distance" to be minimized. (Equivalently, we could maximize log(p_{ab})).

In the next section, we describe how to estimate this function.

2 Probabilistic Distribution

We gathered 40 audio files labeled *queries* from 8 singers: 4 female and 4 male. We showed the singers a list of songs from our database, asked them to select songs that they knew how to sing, and to sing them on any starting pitch they wanted.

We then created manual transcriptions for each query consisting of the notes each person was *supposed to be singing*, given the key they were in. It was usually easy to infer the key they were in because only once did a singer accidentally change key in the middle of a query. For this case we chose the key this singer started in to be the key to transcribe in. We used the audio annotation feature of Audacity (Dominic Mazzoni, 2001) to enter the transcriptions.

We then merged information from the transcribed files with pitch estimates from the queries, computing frequency differences at each 10ms time interval.

Since the singers were allowed to start on any pitch they wanted, often they started on a pitch that was actually in between 2 keys. Because of this, even if the singer had sung perfectly, his or her set of differences between sung and expected pitch values would all be off by as much as a quarter step. To eliminate this tuning problem, we transposed each query such that the median of the differences became zero.

After we collected these sets of normalized pitch differences, we merged them together and created a histogram. The distribution is roughly Gaussian and evenly distributed about zero. This shows us that it is far more likely for a person to be a small amount off when singing, but very unlikely to off by a whole step or more. We then used this data to build a probabilistic distance function that estimates, given 2 pitches (one the expected pitch and one the sung pitch), how likely it is that the singer was aiming for the first pitch but actually sang the second.

To simplify our search, we shifted the histogram by periods of 12 into the [-6, +6] window, and summed the overlapping bins. Therefore we got the probabilistic distribution histogram within one octave.

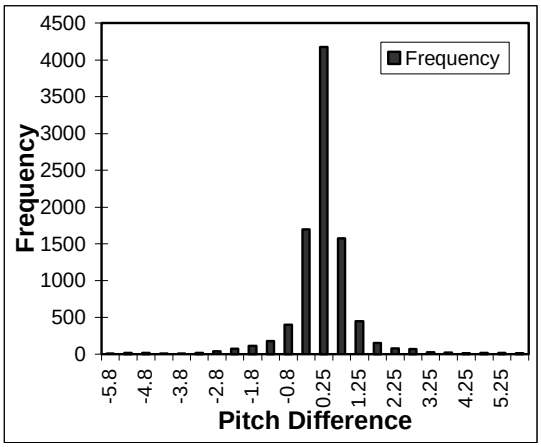

Figure 1. Probabilistic distribution histogram collapsed into one octave. Bins have a width of 0.5 semitones.

3 Dynamic Programming

Although melodic search is inspired by string matching techniques, it has many properties and practical problems that do not exist in string matching.

We investigated several dynamic programming algorithms combining different characteristics. The algorithm that achieves the best results was invented after analyzing important properties of dynamic programming algorithms.

3.1 Edit Distance

For the compared sequence A = $a_1, a_2, \ldots, a_m$ and the query sequence B = $b_1, b_2, \ldots, b_n$, $d_{i,j}$ represents the dissimilarity between $a_1, a_2, \ldots, a_i$ and $b_1, b_2, \ldots, b_j$. The calculation pattern is:

$$d_{i,j} = \min\begin{Bmatrix} d_{i-1,j-1} \\ d_{i-2,j-1} + w(a_{i-1}, b_j) \\ d_{i-1,j-2} + w(a_i, b_{j-1}) \end{Bmatrix} + w(a_i, b_j),$$

$$(1 \le i \le m,\ 1 \le j \le n)$$

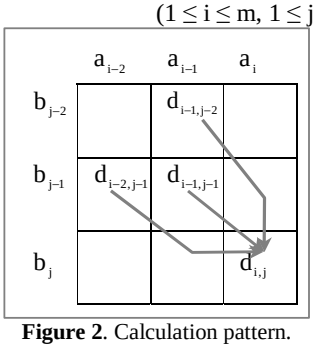

Figure 2. Calculation pattern.

We can think of the algorithm as shrinking the musical contour of either the query or the compared sequence to (locally) double the tempo at certain positions, and then comparing two musical contours exactly. It can also be viewed as assigning a penalty for insertion/deletion.

3.2 Frame-Based Representation

We call the usual note-by-note matching approach *event-based* search to emphasize that notes are discrete events rather than continuous functions of time. Another representation and search strategy we have explored is called *frame-based* search. (Mazzoni & Dannenberg, 2001) Instead of working with discrete notes, the frame-based representation encodes a query rather directly by segmenting the time-varying pitch contour into frames of equal duration. There is no segmentation into notes. This approach is inspired by early speech recognition research (Itakura, 1975) and is also related to the approach of Nishimura, *et al.* (2001). Our search composes melodic contours represented by 10 pitch frames per second.

3.3 Other Important Properties

In the conventional dynamic programming algorithm, there is a penalty for skipping the prefix and suffix of the compared sequence from the original sequence. Alternatively, we can assign no penalty for skipping the prefix and suffix. (McNab, *et al.*, 1996) Thus the initial conditions of the algorithm are:

$d_{i,0} = 0,\ i \geq 0$

$d_{0,j} = d_{0,j-1} + w(\phi, b_j),\ j \geq 1$

We are not very sensitive to absolute pitch, so the pitches of a melody can be shifted, or *transposed*, by any interval. We simply transpose queries into each of 12 possible keys, and ignore octave transpositions. This allows shifts of an octave within a melody without penalty. The pitches of the frames are not quantized but are represented as floating point values. To limit the cost of search, we ignore octaves here as well and search the database 12 times, transposing the query by semitones.

In the same way that pitches can be offset without changing the perceptual quality of a melody, time can also be scaled. We search the database with numerous time-scaled versions of queries (or alternatively, time-scaled versions of the database entries) to cover a reasonable range of tempos. At least one of these time-scaled versions should be a close match to a correct target in the database.

4 Experiment

For the experiment, we collected and processed 598 MIDI files containing popular songs. These include rock songs, folk songs, and TV theme songs, making it easy to invite non-musicians to sing the theme of a song included in the database. The files contain a total of 1,239,138 notes.

The data is processed using the MUSART thematic extractor (Meek & Birmingham, 2001), which locates the 10 most common phrases or melodies from each original file. After processing, there are 5980 entries in the database with an average length of 22 notes. Although this is still a relatively small number, we believe it is large enough to assess the *relative* quality of different melodic comparison and search algorithms. Since the original files now have 10 representative melodies, our search algorithms report the best match to any of the 10 themes as the match score for the original file.

To evaluate the algorithms, we used 37 queries similar to those used to generate the histogram. For each query, we compute a measure of dissimilarity to each entry in the database, and we determine the rank order of the correct database entry for the query. The quality of the algorithm is assessed by counting how many searches return correct songs with a rank order of 1, in the top 10, or in the top 100.

We tested two sets of pitch distance function within the same dynamic programming algorithm, which achieves the best results among all the algorithms we have tested. The first distance function is quite simple. The weight

$$w(a_i, b_j) = \min\begin{Bmatrix} (p_j - p_i) \bmod 12 \\ 12 - ((p_j - p_i) \bmod 12) \end{Bmatrix},$$

p_i represents the pitch of the note a_i in semitones. The range of w is [0, 6].

The second one retrieves the probabilistic distribution histogram and gets the probability depending on the difference from the expected pitch p_i to the sung pitch p_j, so the weight is set be

$$w'(a_i, b_j) = -\log(P(p_j - p_i)).$$

Here P(x) represents the probability when the pitch difference $(p_j - p_i) \bmod 12$ is equal to x mod 12. x ranges from –6 to 6. The range of w' is $[0, +\infty)$.

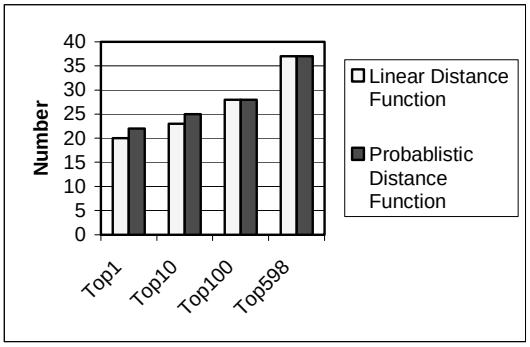

Figure 3. Algorithm comparison chart.

The result shows that under the same conditions, the probabilistic model gets slightly better results than the non-probabilistic model. It is also by far the best among all the algorithms we have tested.

Though the difference between these two algorithms are not large, we believe it is significant since the probabilistic model found all of the melodies found by the non-probabilistic model as well as additional melodies not found by the non-probabilistic model.

5 Conclusions

Melodic similarity is an important subject for many areas of study. In this article, we have demonstrated an approach based on the concepts of probability and maximum likelihood. This approach works better than the use of ad-hoc distance functions when tested on a query-by-humming database problem.

We believe this result is significant for other applications. In particular, it could be used for computer accompaniment, where it is known that certain errors are more likely than others. This gives further support to other research on accompaniment systems that incorporate learning and probabilistic models. (Grubb & Dannenberg, 1997; Raphael, 1999)

Melodic similarity is also important for music understanding and analysis. For example, the work of Cope (Cope, 1996) relies upon melodic comparison to identify similar fragments of scores, and dynamic programming has been used by Rolland (Rolland & Ganascia, 2000) to locate patterns in music.

Finally, melodic similarity has important applications in musicology. (Hewlett and Selfridge-Field, 1998) The use of probabilistic models might allow automated melodic comparison to better approximate perceptual judgments of melodic similarity.

While this work has demonstrated the effectiveness of a probabilistic model of melodic similarity, the model is a simple one. More complex models might take context into account; for example, we have observed that singing accuracy is lower at the beginning of a note and increases somewhat during the course of the note. This could be incorporated into the model by making p_{ab} a function of time position within each note. The model for insertion and deletion penalties should also be refined based on query data.

6 Acknowledgments

Dominic Mazzoni implemented the pitch analysis and transcription software as well as the initial version of the frame-based algorithm. This work is a part of the MUSART project (http://musen.engin.umich.edu/musearts.html) and has benefited greatly from conversations with other project members. This work was supported by the National Science Foundation, Award #0085945.

7 References

Cope, D. (1996). *Experiments in Musical Intelligence* (Vol. 12). Madison, Wisconsin: A-R Editions, Inc.

Dannenberg, R. B. (1984, 1985). "An On-Line Algorithm for Real-Time Accompaniment." *Proceedings of the 1984 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 193-198.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis*: Cambridge University Press.

Grubb, L., & Dannenberg, R. B. (1997). "A Stochastic Method of Tracking a Vocal Performer." *1997 International Computer Music Conference*. San Francisco: International Computer Music Association.

Hewlett, W. and Selfridge-Field, E. (1998). *Melodic Similarity: Concepts, Procedures, and Applications. Computing in Musicology* (Vol. 11). Cambridge: MIT Press.

Hoshishiba, T., Horiguchi, S., & Fujinaga, I. (1996). "Study of Expression and Individuality in Music Performance Using Normative Data Derived from MIDI Recordings of Piano Music." *International Conference on Music Perception and Cognition*. pp. 465-470.

Itakura, F. (1975). "Minimum prediction residual principle applied to speech recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23*, 52-72.

Large, E. W. (1993). "Dynamic programming for the analysis of serial behaviors." *Behavior Research Methods, Instruments, and Computers, 25*(2), 238-241.

Mazzoni, D. (2001). "A Fast Data Structure for Disk-Based Audio Editing." *Proceedings of the 2001 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 107-110.

Mazzoni, D., & Dannenberg, R. B. (2001). "Melody Matching Directly From Audio." *2nd Annual International Symposium on Music Information Retrieval*. Bloomington: Indiana University, pp. 17-18.

McNab, R. J., Smith, L. A., Witten, I. H., Henderson, C. L., & Cunningham, S. J. (1996). "Towards the digital music library: Tune retrieval from acoustic input." *Proceedings of Digital Libraries '96*. ACM.

Meek, C., & Birmingham, W. P. (2001). "Thematic Extractor." *2nd Annual International Symposium on Music Information Retrieval*. Bloomington: Indiana University, pp. 119-128.

Nishimura, T., Hashiguchi, H., Takita, J., Zhang, J. X., Goto, M., & Oka, R. (2001). "Music Signal Spotting Retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming." *International Symposium on Music Information Retrieval*. pp. 211-218.

Raphael, C. (1999). "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models." *IEEE Transactions on PAMI, 21*(4), 360-370.

Rolland, P.-Y., & Ganascia, J.-G. (2000). Musical pattern extraction and similarity assessment. In E. Miranda (Ed.), *Readings in Music and Artificial Intelligence* (pp. 115-144): Harwood Academic Publishers.

Stammen, D., & Pennycook, B. (1993). "Real-Time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm." *Proceedings of the 1993 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 232-235.