MIREX SYMBOLIC MELODIC SIMILARITY AND QUERY BY SINGING/HUMMING

Rainer Typke
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
rainer.typke@musipedia.org

Frans Wiering
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
frans.wiering@cs.uu.nl

Remco C. Veltkamp
Universiteit Utrecht
Padualaan 14
3584 CH Utrecht, Netherlands
remco.veltkamp@cs.uu.nl

Abstract

This submission to the Music Information Retrieval Evaluation eXchange in the Symbolic Melodic Similarity task uses ideas from the system that used the Earth Mover's Distance (EMD) in MIREX 2005. The total weight sums are normalized before applying the EMD, which makes it possible to use a vantage index. A novel way of segmenting is used. Response times are shortened from 14 hours for searching 581 short monophonic incipits to 3 seconds for searching 1000 complete polyphonic pieces of music. This speedup made it possible to put more effort into searching accurately by searching multiple segment sizes at the same time. Therefore, the new method should not only be faster but also more effective.

We submit this algorithm not only for the Symbolic Melodic Similarity task, but also for Query by Singing/Humming. For the latter, we rely on the MIDI files provided by J.-S. Roger Jang (張智星) instead of splitting the given pitch vectors into notes ourselves or working with the wave files.

Keywords: MIREX, symbolic melodic similarity, query by humming/singing.

1. Tasks

At the MIREX competition [1], algorithms from different researchers are compared by letting them solve the same tasks, using the same data. This extended abstract describes a submission to two out of the nine tasks at MIREX 2006.

1.1. Symbolic Melodic Similarity

The task is to retrieve MIDI files that contain material which is melodically similar to a given MIDI query. Half the queries are quantized in rhythm and pitch, and half are only quantized in pitch but not in rhythm. The latter half was created by singing melodies.

[1] MIREX 2006: http://www.music-ir.org/mirex2006

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.
© 2006 University of Victoria

There are three subtasks that differ mainly in the collection of data to be searched:

- Approximately 16,000 incipits from the UK subset of the RISM collection, almost exclusively monophonic. Six queries (three of them quantized).
- 1000 polyphonic Karaoke files. Five queries (two of them quantized; the three sung queries include two versions of the same melody).
- 10,000 randomly chosen MIDI files that were harvested from the Web, most of them polyphonic. Six queries (three of them quantized).

Due to the size of the collections, no ground truth was known in advance. From every participating algorithm, the top ten matches are put into a pool, and human graders judge the relevance.

1.2. Query by Singing/Humming

The Query by Singing/Humming task is split into two subtasks which are strongly influenced by the character of available data. For a collection of 48 MIDI files containing quantized melodies, 2719 sung versions are available as Wave files. For every recording, a pitch vector and a MIDI file were derived. Participants can choose whether to use the audio, pitch vector or MIDI files for answering queries.

1.2.1. Subtask 1: Known-Item Retrieval Task

Based on Prof. Jang's original idea to test for the ability to find the "ground-truth" needle in a collection "haystack".

- Test database: 48 ground-truth MIDIs + $\sim$ 2000 Essen Collection MIDI noise files.
- Queries: 2719 sung queries.
- Evaluation: Mean Reciprocal rank, over top X returns, of the "know-item" ground-truth file for each sung query

1.2.2. Subtask 2: Variants Retrieval Task

Based on Prof. Downie's idea that queries are variants of ground-truth.

- Test database: 48 ground truth MIDIs + $\sim$ 2000 Essen MIDI noise files + 2719 sung queries.
- Queries: 2719 sung queries + 48 ground truth MIDI files.
- Evaluation: Classic precision and recall over X top returns.

2. Indexing

2.1. Splitting polyphonic files into voices

As a preparation for indexing, every MIDI file is split into channels and tracks. It is assumed that every voice is stored in either its own track or channel. In a second step, a skyline algorithm is applied to make each of these extracted voices monophonic.

Even in cases where a voice jumps back and forth between channels or tracks, it can still be searched as long as it stays in one channel or track for at least the minimum segment length (see the next section for a description of segmenting).

2.2. Segmenting

Depending on the collection to be searched, the monophonic voices are split into overlapping segments of varying lengths. If the index size is not a limiting factor (because the collection size is sufficiently small or the computer to be used has enough memory), the segment sizes vary from 5 to 16 consecutive notes. For the collection of 10,000 MIDI files, space on the testing machines was so tight that we could only create segments of sizes 5, 6, and 7. At every note in the voice, a segment of every size begins (unless there are fewer notes left in the piece than necessary for creating a segment of the desired length).

Note that in both cases, the space complexity is $O(N)$, where N is the total number of notes in all pieces to be indexed.

2.3. Vantage indexing

For every segment, the distance to each of a small, fixed set of vantage objects [3] is calculated. As distance measure, the "Proportional Transportation Distance" [5] is used (this is the "Earth Mover's Distance", preceded by a weight normalization such that both weighted point sets have the same total weight).

Later, to answer queries with the vantage index, the distances between the query and each vantage object are calculated, and objects with similar distances to the vantage objects are retrieved from the database. This can be done efficiently with range queries that are supported by B-trees, and it does not involve any expensive EMD calculations except for the comparisons of the query with the small, fixed set of vantage objects.

3. Searching

3.1. Searching an index with many segment lengths

If the index contains segments of length 5 to 16, the query is truncated to 16 notes (if it is longer), and all segments are searched for this possibly truncated query using the vantage index [3]. By not only searching segments with the same length as the query, added or dropped notes, as well as grace notes and other embellishments do not necessarily lead to mismatches but just increase the distance a bit.

In a second step, for the top 50 returned items, the real distance is calculated instead of the estimate that is based on the vantage index. Finally, the top 10 items demanded in the task description are returned.

3.2. Searching an index with just three different segment lengths

If the index only contains segments of length 5, 6, and 7, the query is cut into segments of length 6. For each query segment, the vantage index is searched. Again, for the top 50 returned segments, the real distances are calculated (for each query segment). Finally, these partial results are combined as described in [5]. That is, an optimum combination of query segments is found that match a piece in the same relative positions as their positions within the query, such that the segments' average distance is minimized and the coverage of the query is maximized.

3.3. Considerations for matching documents that are shorter than the query

For the RISM UK collection of musical incipits, one might want to retrieve documents that are shorter than the query, while this case is very unlikely for a collection consisting of complete pieces. To support this possibility for the RISM subtask, even though the incipits are split into segments of 5 to 16 consecutive notes, the query is not just capped at 16 notes as described above, but split into segments of varying sizes from 5 consecutive notes up to either 16 or the length of the query, whatever is lower. That way, shorter incipits can be matched to parts of the query, but the whole query can still be matched to longer incipits in one comparison.

Currently, perfect matches of a whole incipit that is shorter than the query lead to a distance of zero, and so do perfect matches of the whole query with parts of a longer incipit. Assigning different distances (a lower distance to the latter case, where more notes match), would be an improvement, but one would need to investigate how big such a difference should be in order to agree with human ideas of similarity.

3.4. The Query by Singing/Humming task

We submit our algorithm not only for Symbolic Melodic Similarity but also for the Query by Singing/Humming task. However, we do not analyze the wave or pitch vector files, but instead work only with the MIDI files provided by J.-S. Roger Jang along with the collection of queries. By doing

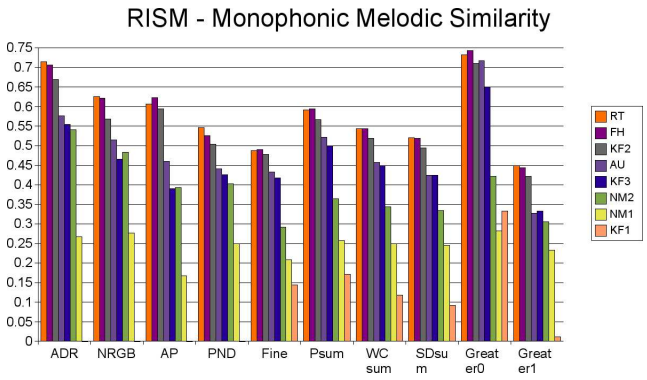

Figure 1. Task I: RISM Overall Summary. See Section 4.1.2 for an explanation of the measures. The methods are: RT - the method described in this paper; FH - an editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [2], KF is a hybrid distance measure by Klaus Frieler and Daniel Müllensiefen, AU is Alexandra Uitdenbogerd's Start-Match Alignment technique, and NM is the geometric "P3" algorithm by Kjell Lemström, Niko Mikkilä, Veli Mäkinen and Esko Ukkonen. For a more detailed description of these methods, see the MIREX abstracts, available from http://www.music-ir.org/mirex2006/index.php/Symbolic_Melodic_Similarity_Results

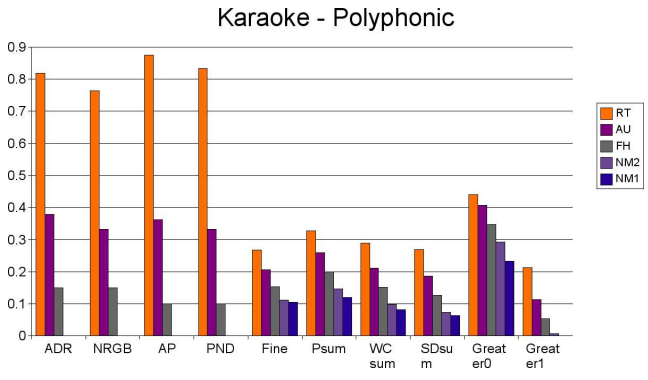

Figure 2. Task IIa: Karaoke Overall Summary

so, we avoid the need to change the algorithm at all, but any error in his conversion from Wave to MIDI will reduce our performance.

For this task, we submit the algorithm with both indexing variants – with segments of lengths 5 to 16 and with segments of lengths 5, 6, and 7. We treat the queries the same way as for the Symbolic Melodic Similarity task, that is, if we have many segment lengths, we cut the query at 16 notes, while for just three different segment lengths, we segment the query into segments of length 6. Since it is known that in Jang's collection, the queries match the database items only at the beginning, we index only the first 25 notes of every item.

The variant with fewer segment lengths needs less space for the index (only three segments start at every note instead of eleven), but requires more computing time for answering queries since there are multiple segment searches, and their results need to be consolidated into one overall result.

4. Results, Analysis

4.1. Symbolic Melodic Similarity

4.1.1. Building a ranked list from relevance scores

The raw ground truth data consisted of a rough and a fine relevance score for every item that was returned by an algorithm. For the rough score, a scale of "very similar", "somewhat similar", and "not similar" was used, while the fine score was just a number between 0 and 10. For the polyphonic tasks, where algorithms returned excerpts of MIDI files but not whole MIDI files, separate relevance scores were collected for each excerpt, even if there were multiple excerpts from the same MIDI file.

For some measures, an ordered list of relevant items is necessary. These ordered lists were created as follows from the collected relevance scores:

- Calculate average scores for every MIDI file; these averages were taken from three human graders if a MIDI file was returned by only one algorithm, or by a multiple of three people if multiple algorithms returned the same polyphonic MIDI file.
- For each query, order the matches first by the rough and, in case of ties, by the fine score.
- Group together matches with the same average rough score.
- Only include items with average rough scores of better than "somewhat similar".
- If the resulting list is longer than 10, remove whole groups at the end until at most 10 items remain; there was one exception where the top group had 11 "very similar" items.

4.1.2. Measures

The following measures were used (these abbreviations are used in Figures 1, 2, and 3):

- ADR = Average Dynamic Recall [4].
- NRGB = Normalized Recall at Group Boundaries.
- AP = Average Precision (non-interpolated).
- PND = Precision at N Documents.
- Fine = Sum of fine-grained human similarity decisions (0-10).
- PSum = Sum of human broad similarity decisions: NS=0, SS=1, VS=2.

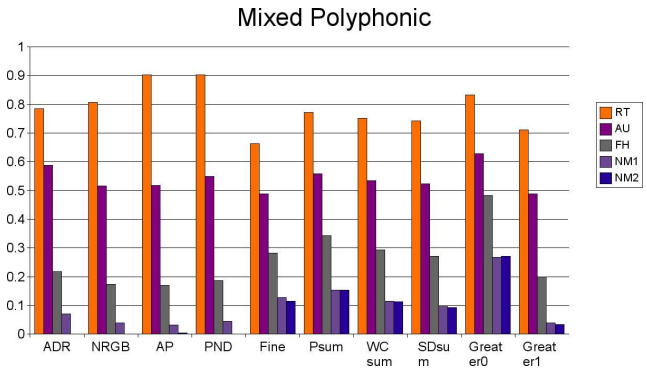

Figure 3. Task IIb: Mixed Polyphonic Overall Summary

- WCsum = 'World Cup' scoring: NS=0, SS=1, VS=3 (rewards "very similar").
- SDsum = 'Stephen Downie' scoring: NS=0, SS=1, VS=4 (strongly rewards "very similar").
- Greater0 = NS=0, SS=1, VS=1 (binary relevance judgement).
- Greater1 = NS=0, SS=0, VS=1 (binary relevance judgement using only "very similar").

All measures are normalized such that they lie in the range from 0 to 1.

4.1.3. Monophonic Task

For the monophonic task, there were no significant performance differences between our method and the editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [2], [1]. When looking at the actual result lists – they are available at http://rainer.typke.org/mirex06.0.html – the main difference between the results of these two methods seems to be that the latter is more likely to retrieve rather short matches (compare, for example, http://rainer.typke.org/qr3-fh.0.html and http://rainer.typke.org/qr3-rt.0.html). In some cases, this might have lead to a lower average precision or average dynamic recall, like for example in the case of http://rainer.typke.org/qr6-fh.0.html, where short matches pushed nice longer ones down to lower ranks.

The other methods performed worse than ours and Ferraro's/Hanna's, no matter which measure is used.

4.1.4. Polyphonic Tasks

For both polyphonic tasks, our method outperforms the other methods.

Besides the obvious difference in the number of notes that can sound at the same time, the polyphonic collections also differ in other ways from the monophonic RISM collection:

- Both polyphonic collections were random sets of files that were harvested from the Web. Because of this, the encoding quality was not as homogeneous as for the RISM collection. Some files in the polyphonic collection were not syntactically correct MIDI files.
- While the RISM collection was created from plaine&easie code and therefore rhythmically quantized, the polyphonic collections contained both quantized music and renditions of performances, where neither onset times nor note durations were exactly the same as what one would find in a written score.

When looking at Figures 2 and 3, it is very noticeable that the six rightmost measures, which were not based on the ranked lists described in Section 4.1.1, indicate a much worse performance for the Karaoke task for all algorithms. The reason is that the Karaoke collection was much smaller (1000 items instead of the 10,000 items in the mixed collection) and therefore contained fewer good matches to begin with. Even an ideal algorithm can therefore not reach a score of 1 for measures such as "Fine" for the Karaoke collection. The four measures on the left side compare the algorithms' outputs with the ground truth lists.

4.2. Query by Singing/Humming

No algorithm that relied on the provided MIDI files turned out to be very successful. These MIDI files contained many incorrectly recognized additional notes with more or less random pitches.The most successful algorithm that used the MIDI files provided by Jang, NM, performed an exhaustive depth-first search trying to scale the pattern note-by-note to fit the song, with costs applied to local time-scaling, note duration changes and pitch-shifting (note that this is not the same algorithm as the "symbolic" NM algorithm). Our algorithm performed relatively poorly because it looked for matches for *all* query notes, including the random added notes, without the possibility of reducing the importance of individual query notes. The NM algorithm is able to selectively ignore query notes that do not fit any note in a matching piece. For a similar effect with our method, we would need to skip the weight normalization step and therefore work with a pure EMD. However, this would reduce the effectiveness of vantage indexing – it would remove the guarantee that indexing does not lead to false negatives.

Many of the successful algorithms used symbolic approaches; for example, Xiao Wu and Ming Li use a transcription to notes for filtering out candidates for matches, followed by a final scoring on the frame level. Christian Sailer as well as Ernesto Lopez and Martín Rocamora do all matching in the symbolic domain after transcribing the audio signal to notes. An important reason for such hybrid approaches outperforming our method was that they used their own elaborate methods for transcribing the audio signal into notes.

	Task I (MPR)	Task II (Mean Precision)	Uses audio or MIDI
XW1 (Xiao Wu and Ming Li)	0.926	no entry	audio
XW2	0.900	no entry	audio
RJ (Roger Jang and Nien-Jung Lee	0.883	0.926	audio
RL (Ernesto Lopez and Martín Rocamora)	0.800	no entry	audio
NM	0.688	0.722	MIDI
CS1 (Christian Sailer)	0.568	0.587	audio
RT2	0.390	0.401	MIDI
CS3	0.348	0.415	audio
AU2	0.288	0.238	MIDI
CS2	0.283	0.649	audio
FH	0.218	0.309	MIDI
AU1	0.205	0.163	MIDI
RT1	0.196	0.468	MIDI

Table 1. Overall results for QBSH. NM, RT, AU, and FH use Jang's MIDI files, while the other methods do their own note transcription (except for RJ's pure audio approach). RT2 segments the queries, while RT1 does not. There seems to be a strong correlation between using the provided MIDI files and poor overall performance. For detailed descriptions of the various methods, see the MIREX abstracts at http://www.music-ir.org/mirex2006/index.php/QBSH:_Query-by-Singing/Humming_Results

Maybe one can learn from this experience that symbolic approaches can be valuable for searching large audio databases because they can make searches very efficient without necessarily ruining the quality of results, especially if the last scoring step is again done in the audio domain and therefore avoids the problems that note transcription introduces. After reducing the number of candidates for matches by using an efficient symbolic method, one can afford more expensive, but effective audio analyses since they only need to be done for a small number of items.

5. Acknowledgments

Many thanks to IMIRSEL for the huge amount of effort spent in running the evaluation, to Anna Pienimäki for co-leading the Symbolic Melodic Similarity task and creating the queries, to Maarten Grachten for his help with converting RISM incipits to MIDI, to RISM UK for allowing us to use their incipits, and to Niall O'Driscoll from Alexa for providing Musipedia.org with free access to Alexa's web search product (the test collection of polyphonic MIDI files is a subset of the Musipedia web search data set). For the Query by Singing/Humming task, we are grateful for Stephen Downie and J.-S. Roger Jang (張智星) leading the task, and for the data collection of wave files, pitch vectors, and MIDI files provided by the latter.

References

[1] P. Ferraro and C. Godin. "An Edit Distance Between Quotiented Trees", *Algorithmica*, 36:1–39, 2003.

[2] P. Ferraro and P. Hanna. "Symbolic Melodic Similarity," *MIREX 2006 abstract*.

[3] R. H. van Leuken, R. C. Veltkamp, and R. Typke. "Selecting vantage objects for similarity indexing," *International Conference on Pattern Recognition (ICPR)* 2006, Hong Kong.

[4] R. Typke, R. C. Veltkamp, and F. Wiering. "A measure for evaluating retrieval techniques based on partially ordered ground truth lists", *International Conference on Multimedia & Expo (ICME)* 2006, Toronto, Canada.

[5] R. Typke, F. Wiering, and R. C. Veltkamp. "Transportation distances and human perception of melodic similarity," *ESCOM Musicæ Scientiæ (to appear)*.