Melodic track identification in MIDI files

David Rizo, Pedro J. Ponce de León, Antonio Pertusa and **José M. Iñesta**
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, Spain

drizo,pierre,pertusa,inesta@dlsi.ua.es
http://grfia.dlsi.ua.es

Abstract

The objective of this work is to find the melodic line in MIDI files. Usually, the melodic line is stored in a single track, while the other tracks contain the accompaniment. The detection of the track that contains the melodic line can be very useful for a number of applications, such as melody matching when searching in MIDI databases. The system was developed using WEKA. First, a set of descriptors from each track of the target melody is extracted. These descriptors are the input to a random forest classifier that assigns a probability of being a melodic line to each track. The tracks with a probability under a given threhold are filtered out, and the one with the highest probability is selected as the melodic line of that melody. Promising results were obtained testing different MIDI databases.

Introduction

The goal of this work is to find the melodic line track in a MIDI[1] file. The standard MIDI file format is a representation of music designed for replay through electronic instruments (Uitdenbogerd & Zobel 1999). The melodic line (also called melody voice) is the leading part in a composition with accompaniment. A MIDI file is structured into tracks. Usually, each track contains the notes played by a single instrument, and the melodic line is often stored in a single track.

The detection of the melodic line can be useful, for example, to help algorithms that detect similarities between two songs (melody matching (Uitdenbogerd & Zobel 1999)) or those that search songs in a MIDI database by humming the voice melody (Ghias *et al.* 1995).

The literature about this topic is quite poor. Several papers aim to extract the melodic line from audio files (Berenzweig & Ellis 2001)(Eggink & Brown 2004). Ghias et al. (Ghias *et al.* 1995) built a system to process MIDI files extracting something similar to the melodic line using simple heuristics not described in their paper and ignoring the MIDI percussion channel. Some authors (Marsden 1992) split polyphonic[2] songs into parts, usually detecting patterns as repetitions (Cambouropoulos 1998)(Lartillot 2003)(Meudic 2003).

In (Uitdenbogerd & Zobel 1998), four algorithms were developed to detect the melodic line in polyphonic MIDI files, assuming that the melodic line is monophonic. These algorithms were applied in a latter work for melody matching (Uitdenbogerd & Zobel 1999). No track information (except ignoring the percussion events) was taken into account.

The goal of our work is not to extract a monophonic line from a polyphonic score, but to detect which one of the MIDI tracks corresponds to the main melody, due to the melodic line of a song is often stored in a single track, specially in popular music.

The concept of a melody voice and accompaniment must be defined prior to be able to select the melodic track from a MIDI file. There are some features in a melodic track that, at first sight, seem to be definitive to identify it, like having high pitches or being monophonic[3]. Unfortunately, it is common to find a melodic line that has not the highest average pitch of the song, or that contains some chords.

To overcome these problems, a classifier that learns what is a melodic track and what is not was utilized. The WEKA (Ian H. Witten 1999) toolkit was chosen to build the system, and it was extended to read the track descriptors proposed in section directly from MIDI files.

The training set consists of all the tracks of the tested songs. Each track is represented by an instance, and the attributes of each instance are described in . Several experiments were performed to choose a classifier among the ones implemented in WEKA. The random forest classifier (Breiman 2001) using 10 trees in each forest and considering 5 atributes yielded the best results, so the experiments described in section were done using this classifier.

Therefore, a set of descriptors is extracted from each track of a target melody, and these descriptors are the input to a classifier that assigns a melodic line probability for each track. The tracks with a probability under a given threhold

[1] www.midi.org

[2] In a polyphonic song there can be several notes sounding simultaneously.

[3] In a monophonic track there can be only one note sounding simultaneously

are filtered out, and the one with the highest probability is selected as the melodic line of that melody.

Methodology

MIDI Track characterization

The content of a track is characterized by a vector of statistical descriptors based on descriptive statistics that summarize track content information. This kind of statistical description of musical content is sometimes referred to as *shallow structure description* (Pickens 2001). A set of 12 descriptors has been defined, based on several categories of features that assess melodic and rhythmic properties of a music sequence, as well as track properties. These descriptors showed evidence of statistical significance when comparing their distribution for melody and not-melody tracks. Other considered descriptors did not show significant difference when comparing their per-class distribution, so they have not been used in the experiments described.

For training and testing purposes, each track is labelled as being a melody track or not. A list of the descripors used in this work is shown below.

- Track descriptors
  - Normalized track length
  - Polyphony rate
  - Occupation rate
- Pitch descriptors
  - Highest normalized pitch
  - Lowest normalized pitch
  - Average normalized pitch
- Note duration descriptors
  - Highest normalized duration
  - Lowest normalized duration
- Interval[4] descriptors
  - Highest normalized absolute interval
  - Lowest normalized absolute interval
  - Average normalized absolute interval
  - Normalized number of distinct intervals

Track duration and note duration descriptors are computed as the ratio between the duration value in ticks and the MIDI file resolution. This way, durations are expressed as a number of beats to make them independent from the midifile resolution. Pitch, note duration, interval and track length descriptors are normalized using the formula $(value - min)/(max - min)$, where $value$ is the descriptor to be normalized, and min and max are respectively the minimum and maximum value for this descriptor for all the tracks of the target midifile. For pitch descriptors, the maximum value is fixed at 127 (the highest possible pitch value in any MIDI file), and the minimum is set to 0 (the lowest possible pitch value).

[4]Distance in pitch between two consecutives notes

In order to characterize the degree of polyphony in a track, the polyphony rate is defined as the ratio between the number of ticks in the track where two or more notes are active, and the track duration in ticks. The occupation rate descriptor accounts for the percentage of the track length that is occupied by notes, and is defined as the ratio between the number of ticks where at least one note is active and the track length in ticks.

Pitch descriptors are the highest, lowest and average normalized pitch in the track. Note duration properties are described by the highest and lowest normalized note durations found in a track. The interval descriptors summarize information about the difference in pitch between consecutive notes. Pitch interval values are either positive or zero, when the first pitch is lower or equal to the second pitch, or negative, when the first pitch is higher than the second one. However, pitch interval information is collected as absolute values, and it is summarized as the highest, lowest and average normalized values. Finally, the number of distinct absolute interval values is counted and normalized among tracks.

To summarize, information about track content and pitch distribution, note duration distribution, and absolute interval distribution in a track is provided to describe the content of a MIDI track as a vector of real numbers, normalized between 0 and 1. This is the representation used to train the random forests classifier, as described in the next section.

The random forest classifier

Random forests are a combination of tree predictors that use a random selection of features to split each node. This classifier yields error rates that compare favorably to techniques like Adaboost, but are more robust with respect to noise. The forest consists of K trees. Each tree is built using CART (Duda, Hart, & Stork 2000) methodology to maximum size and do not prune. The number F of randomly selected features to split on at each node is fixed for all trees. After growing the trees, new samples are classified by each tree and their results are combined, giving as a result a membership probability for each class. In our case, this is simply the probability of being a melodic line track. In the experiments presented in section , $K = 10$ trees are used, and $F = 5$ features are randomly selected to split each tree node.

Track selection

Once a model discriminates between melodic tracks and non-melodic tracks, the problem is to select a track as the melodic line of the song. There are MIDI files that contain more than one track suitable to be classified as melodic line. The same way, as usually happens in classical music, some songs do not have a well-defined melodic line, like in some piano sequencies. The algorithms proposed in (Uitdenbogerd & Zobel 1998) seem more suitable to deal with that cases.

In this work, only one track is selected as the melodic line. The goal is that this track will really be a melodic line, even if there is more than one possible melodic track in the MIDI file. Therefore, given a melody, all its tracks are classified and their probability p of being a melodic line is stored.

Track	p
1	0.4
2	**0.8**
3	0.7

Table 1: Melodic line probability for a sample song. Each row corresponds to a track.

For example, suppose a song has three tracks, and the results of classifying them are those given in table 1. In this sample, the track *2* will be selected as the melodic track. If there are no tracks in a MIDI file with a probability p greater than 0.75, no track is chosen and an error is reported.

Experiments

Test corpora

Six corpora (see table 2) were created due to the lack of existing databases for this task. There are a lot of MIDI files available on internet, but it is difficult to find melodies labelled with their melodic line. Two corpora with jazz files, another two with classical music pieces where there is a clear melodic line, and two more for sung popular music in karaoke (.kar) format were utilized for the experiments.

These files were downloaded from various internet sources. From thousands of available files, only those with some track whose name in lowercase is in the set {*melody, melodie, melodia, vocal, chant, voice, lead, lead vocal, canto*} were selected. These tracks were labelled as melodic lines. Remaining tracks were labelled as non-melodic tracks. The MIDI percussion tracks (channel 10) were removed for the experiments.

Corpus name	Style	Number of files
CLAS200	Classical	200
RB200	Jazz	200
KAR200	Popular	200
CLAS	Classical	51
AMNZ	Jazz	998
KAR	Popular	1360

Table 2: Corpora

Results

Different experiments were carried out. Some WEKA classifiers were tested, and the random forests yielded the best results. First, the capability of the random forests to classify between melodic and non melodic tracks was measured. Then, the system was tested with different train/test corpora to evaluate the accuracy of the system.

Instance classification Given a set of instances (tracks), this experiment outputs the percentage of correctly classified instances. A ten-folded cross-validation experiments were performed to estimate the accuracy of the method. The results for the CLAS200, RB200 and KAR200 corpora are shown in table 3.

Corpus	Correctly Classified Instances percentage
CLAS200	99.5
RB200	97.9
KAR200	94.6

Table 3: Instance classification results

Corpus	Correcty selected tracks percentage
CLAS200	99
RB200	99.5
KAR200	87.5

Table 4: Melody track selection, leave-one-out results

Melodic track selection Three different kind of experiments were performed. The first one uses the leave-one-out scheme for each one of the 200 files corpora. The second one evaluates the system robustness across different corpora. Finally, a cross-validation experiment (4-folds) was done to test the overall accuracy.

Leave-one-out experiments were also performed, and results are shown in table 4.

Another experiment was performed crossing the different musical styles corpora. This was done training with two styles and classifying the left one. The results are described in table 5.

Training corpora	Test corpus	% Success
CLAS200+RB200	KAR200	58.5
CLAS200+KAR200	RB200	81.5
KAR200+RB200	CLAS200	96

Table 5: Melody track selection across styles

The results in table 5 show that when the random forest was trained with the classical and jazz corpora, the popular music was not successfully classified. To overcome this problem, a new training set was built with the three 200 files corpora, using it to test the another different corpora. The results are detailed in table 6. When the training set contains the three styles the results are improved over the ones obtained when training with only two styles.

Finally, the same three corpora were merged into a 600 songs set to perform a cross-validation experiment. The entire set was divided into four subsets of songs. The experiments shown in table 7 were done using three subsets for training and the left one for test. The average success rate was 0.93.

Conclussions and future work

The goal of this work is to find the melodic line track in a MIDI file. The WEKA toolkit was chosen to develop the system. A set of descriptors was extracted from each track of a target melody. These descriptors are the input to a random forest classifier that determines whether a track is a melodic line or not. The tracks classified as melodic lines are then

Training corpora	Test corpus	% Success
CLAS200+RB200+KAR200	KAR	87
CLAS200+RB200+KAR200	AMNZ	97
CLAS200+RB200+KAR200	CLAS	92

Table 6: Melody track selection training with all styles

	Success rate
Fold 1/4	0.92
Fold 2/4	0.93
Fold 3/4	0.94
Fold 4/4	0.93
Average	0.93

Table 7: 4-fold cross-validation results.

selected, and the one with the highest probability is finally labelled as the melodic line of that song.

The experiments yielded promising results using different databases. Unfortunately, the results could not be compared with other systems because of the lack of similar works.

This system could be used to convert a MIDI files database into a melodic lines database. The queries over the obtained database should be easier on a content based search over the melodic line, like those made by humming the main melody.

The experiments show that enough training data of each style is needed in order to successfully detect the melodic track in each style. The melodic line is hard to detect in symphonic music, where there is not a standalone track that corresponds to the melodic line. Instead, the melodic track changes as the song develops, because the main voice instrument changes from section to section. To overcome this problem, we plan to segment the whole song and locate the melodic track in each segment.

A melodic line is usually a monophonic sequence of notes. However, some melodies encoded as MIDI file tracks have some degree of overlapping between consecutive notes. This is often the case with real-time sequenced tracks, where melodies are recorded as they are played by a musician. A preprocessing of these tracks could be done to avoid this note overlapping, detecting the tracks that have this problem and making them monophonic. This would probably increase the accuracy of the system.

Acknowledgements

This work was supported by the projects Spanish CICyT TIC2003–08496–C04, partially supported by EU ERDF, and Generalitat Valenciana GV043–541.

References

Berenzweig, A., and Ellis, D. 2001. Locating singing voice segments within music signals. In *Proceedings of the IEEE workshop on Applications on Signal Processing to Audio and Acoustics (WASPAA)*.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Cambouropoulos, E. 1998. *Towards a general computational theory of musical structure*. Ph.D. Dissertation, Faculty of music and Department of Artificial Intelligence.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2000. *Pattern Classification*. John Wiley and Sons.

Eggink, J., and Brown, G. J. 2004. Extracting melody lines from complex audio. In *5th International Conference on Music Information Retrieval (ISMIR)*.

Ghias, A.; Logan, J.; Chamberlin, D.; and Smith, B. C. 1995. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, 231–236.

Ian H. Witten, E. F. 1999. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann.

Lartillot, O. 2003. Perception-based advanced description of abstract musical content. In Izquierdo, E., ed., *Digital Media Processing for Multimedia Interactive Services*.

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

Meudic, B. 2003. Automatic pattern extraction from polyphonic midi files. In *Communications in 10 Journ'ees d'Informatique musicale*.

Pickens, J. 2001. A survey of feature selection techniques for music information retrieval. Technical report, Center for Intelligent Information Retrieval, Departament of Computer Science, University of Massachussetts.

Uitdenbogerd, A. L., and Zobel, J. 1998. Manipulation of music for melody matching. In *Multimedia '98: Proceedings of the sixth ACM international conference on Multimedia*, 235–240. ACM Press.

Uitdenbogerd, A., and Zobel, J. 1999. Melodic matching techniques for large music databases. In *Multimedia '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, 57–66. ACM Press.