A probabilistic approach to melodic similarity

José F. Bernabeu, Jorge Calera-Rubio, José M. Iñesta, David Rizo

Dept. Lenguajes y Sistemas Informáticos, University of Alicante, Spain
{jfbernabeu,calera,inesta,drizo}@dlsi.ua.es

Abstract. Melodic similarity is an important research topic in music information retrieval. The representation of symbolic music by means of trees has proven to be suitable in melodic similarity computation, because they are able to code rhythm in their structure leaving only pitch representations as a degree of freedom for coding. In order to compare trees, different edit distances have been previously used. In this paper, stochastic k-testable tree-models, formerly used in other domains like structured document compression or natural language processing, have been used for computing a similarity measure between melody trees as a probability and their performance has been compared to a classical tree edit distance.

1 Introduction

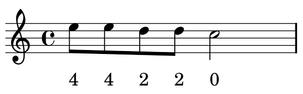
Music pieces can be represented by symbolic structures such as strings or trees containing the sequence of notes in the melody. A melody has two main dimensions: rhythm (duration) and pitch. In linear representations, both pitches and durations are coded by explicit symbols, but trees are able to implicitly represent time in their structure (the shorter a note the deeper it is in the tree), making use of the fact that note durations are multiples of basic time units in a binary (sometimes ternary) subdivision. This way, trees are less sensitive to the codes used to represent melodies, since only pitch codes are needed to be established and thus there are less degrees of freedom for coding.

In this paper, the problem of comparing symbolically encoded (e.g. MIDI or MusicXML) musical works is addressed. For it, we will represent melodies as trees and we will measure their similarity as a probability of being the same melody. In order to compare trees, different edit distances have been previously used [1, 2]. Here we will learn probabilistic k-testable tree models [3], a generalization of the k-gram models, that are easy to infer from samples and allow incremental updates. They can be used for data categorization if a model is inferred for each class and the new samples are assigned a probability by each model taking a maximum likelihood decision. This approach is going to be tested in this work for cover version identification as a benchmark to study its feasibility, but it can be applied to other music information retrieval (MIR) scenarios with little, if any, adaptation.

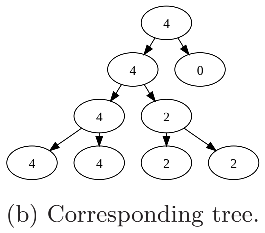
2 Melody tree representation

For representing the note pitches in a monophonic melody s as a string, symbols σ from a pitch representation alphabet Σ_p are used: $s \in \Sigma_p^*, s = \sigma_1\sigma_2...\sigma_{|s|}$. In this paper, the interval from the tonic of the song modulo 12 is utilized as pitch descriptor (Fig. 1(a)): $\Sigma_p = \{p \in \mathbb{N} \mid 0 \leq p \leq 11\} \cup \{\text{`}-\text{'}\}$. This way, in 'G Major', any pitch 'G' is mapped to 0. This alphabet permits a transposition invariant representation. Rests are represented by a special label '–'.

In the proposed approach, each melody bar is represented by a tree, $t \in T_{\Sigma_p}$ (the set of trees that can be made with the labels in Σ_p). The level of a node in the tree determines the duration it represents (see an example in Fig. 1(b)). The root (level 1) represents the duration of the whole bar, the two nodes in level 2 the duration of the two halves of a bar, etc. In general, for a binary meter, nodes at level i represent duration of a $1/2^{i-1}$ of a bar ($1/3^{i-1}$ for a ternary meter). Therefore, during the tree construction, nodes are created top-down when needed and guided by the meter, to reach the appropriate leaf level to represent a note duration (notes are split to accomodate node durations). At that moment, the corresponding leaf node is labeled with the pitch representation symbol, $\sigma \in \Sigma_p$. Once the tree has been built, a bottom-up propagation of the pitch labels is performed to label all the internal nodes. The rules for this propagation are based on a melodic analysis [4]. All the notes are tagged either as *harmonic tones* for those belonging to the current harmony at each time, or as *non-harmonic tones* for those ornamental notes. Harmonic notes have always priority for propagation and when two harmonic notes share a common father node, propagation is decided according to the metrical strength of the note (the stronger the more priority), depending on its position in the bar and the particular meter of the melody. Note that each bar may have a diferent time signature. Notes have always higher priority than rests. Eventually, all the internal nodes are labeled, yielding the tree t_i that codes the i-th bar of the melody (see Fig. 1(b)). This process is repeated for all the bars in the melody.



(a) Original melody. The figures below the score are the labels in Σ_p (interval from tonic mod 12).



(b) Corresponding tree.

Fig. 1: Tree representation of a single measure. Labels are pitch classes relative to tonic (C major in this case).

At this point, all the bar trees: $t_1, t_2, ..., t_{|M|}$, where $|M|$ is the length of the melody in bars, are linked to a common root, building up a forest, $\sigma(t_1 t_2 ... t_{|M|}) \in$

T_{Σ_p}, where that common root is labeled with the root of the first tree, corresponding to the first harmonic tone of the melody, after the melodic analysis performed for the label bottom-up propagation.

3 Methods

Stochastic models based on k-grams predict the probability of the next symbol in a sequence depending on the $k-1$ previous symbols. They have been extensively used in natural language modeling and also some works on MIR [5, 6].

From a theoretical point of view, k-gram models can be regarded as a probabilistic extension of locally testable languages. A string language $\mathcal{L}$ is locally testable if every string w can be recognized as a string in $\mathcal{L}$ just by looking at all the substrings in w of length at most k. These models are easy to learn and can be efficiently processed [7].

In the case of locally testable tree languages, as described by Knuutila [8], the concept of k-fork, f_k, plays the role of the substrings and the k-root, r_k, and k-subtrees, s_k, play the role of prefixes and suffixes. For any $k > 0$, every k-fork contains a node and all its descendents lying at a depth smaller that k. The k-root of a tree is its shallowest k-fork and the k-subtrees are all the subtrees whose depth is smaller than k. So, to infer a *deterministic finite-state tree automaton* (DTA) $A = (Q, \Sigma, \Delta, F)$ that recognizes a k-testable tree language T from a sample S is computed as:

- ▶ $Q = r_{k-1}(S) \cup r_{k-1}(f_k(S)) \cup s_{k-1}(S)$ (set of states);
- ▶ $\Sigma = \Sigma_p = \{p \in \mathbb{N} \mid 0 \leq p \leq 11\} \cup \{\text{`}-\text{'}\}$ (alphabet);
- ▶ $F = r_{k-1}(S)$ (subset of accepting states);
- ▶ add to Δ (set of transitions) a transition
  - $\delta(\sigma, t_1, ..., t_m) = \sigma(t_1, ..., t_m)$ for every $t = \sigma(t_1, \ldots, t_m) \in s_{k-1}(S)$;
  - $\delta(\sigma, t_1, ..., t_m) = r_{k-1}(\sigma(t_1, ..., t_m))$ for every $t = \sigma(t_1, \ldots, t_m) \in f_k(S)$.

In a stochastic classification task, a sample is assigned to the class that maximizes the probability of generating it. For a new melody represented as a tree to be classified in a particular class of melodies, we need to infer a probabilistic DTA for each class, C_j, from correctly classified melodies. For this purpose, one should note that the likelihood of the sample is maximized if the stochastic model assigns to every tree in the sample a probability equal to the relative frequency of the tree in the sample [9]. So, we only need to count the number of k-forks, $(k-1)$-subtrees and $(k-1)$-roots. If we store the probabilities as the quotient of two counters, the automaton can be updated incrementally (one for transitions in Δ and other for states in Q).

Once the probabilistic DTAs for the different classes have been inferred and the probabilities estimated (see [3] for the details), a melody M is classified in the class $\hat{C}$ that maximizes the likelihood

$$\hat{C} = \arg\max_j \; l(M|C_j) \tag{1}$$

where the likelihood of the melody for each class is computed with

$$l(M|C_j) = \rho^{[2]}(\sigma|C_j) \prod_{i=1}^{|M|} \pi^{[k]}(t_i|C_j) \tag{2}$$

where $\rho^{[2]}(\sigma|C_j)$ is the probability of the root of the forest, σ, for the model $k = 2$, for class C_j. Using $k = 2$ means that only the label of that common root is utilized, this way avoiding the probability of a given melody to belong to a class depends on its number of bars (its length). The other term in the equation is the conditional probability of the new melody for class C_j, calculated by multiplying the probabilities $\pi^{[k]}$ of all the bars of a given melody where $\pi^{[k]}(t_i|C_j)$ is the product of the probabilities of the transitions utilized to process the tree t_i (i-th bar of the melody M) for the used model k.

The problem occurs when parsing a tree the system finds a transition not seen in the training set. To solve this problem a smoothing method is applied. The approach used in this paper is the *backing-off* technique.

Backing-off methods have been extensively studied for string models [9]. The underlying idea is to discount some probability mass to the seen events and distribute it among the unseen events. For it, models with $1 \leq k \leq K$ are needed. The smaller the k value, the more general the model is. The aim is always to compute the probability of a transition with the K model. If it does not have the needed transition then the $k-1$ model is utilized. If necessary, this process is repeated until the $k = 1$ model is used. This is a base model that never assigns null probabilities and therefore is able to recognize any tree through its component nodes (see [3] for the details).

4 Experiments and results

In our experiments, we try to identify a problem melody using a set of different variations played by musicians. For that, we use a corpus consisting of a set of 420 monophonic 8-12 bar themes of 20 worldwide well known tunes of different musical genres[1]. For each song, a canonic version was created using a score editor and synthesized. The audio files were given to three amateur and two professional musicians who listened to the songs and played them on MIDI controllers (real-time sequencing them) 20 times with different embellishments and capturing performance errors. This way, for each of the 20 original scores, 21 different variations have been built (all of them with 4/4 meter signature).

The results using the proposed maximum likelihood technique were compared to those obtained for the same data in [10], where for each target melody, classical tree edit distances to all the prototypes were computed and the 1-NN rule was applied for taking the decision. The edit distance was computed according to Selkow's algorithm [2] with insertion, deletion and substitution costs set to 1.

[1] The MIDI data set is available upon request to the authors.

A 3-fold cross-validation scheme was carried out to perform the experiments, obtaining average success rates and dispersions $(\max - \min/4)$. Values for $K = \{2, 3, 4\}$ were utilized (see Table. 1).

Tree edit dist.	$K = 2$	$K = 3$	$K = 4$
82.0 ± 0.2	85.7 ± 0.5	93.3 ± 0.5	86 ± 2

Table 1: Average success rates for the three-fold experiments and comparison with tree edit distances.

These results show that stochastic k-testable methods clearly improve the results of the classical tree edit distance. Note that the dispersions calculated from the cross-validation experiments are very low, so the improvement is very significant, specially for $K = 3$. $K = 2$ is a more general, and therefore less discriminative, model and $K = 4$, although a priori more powerful, is conditioned for the low cardinality of the training set. The higher the number of different probabilities to be inferred, the more data needed, so the melodies available in our variation data set are not enough for this model to outperform $K = 3$.

Fig. 2 shows the success rates taking into account the first n most probable classes (songs) for different values of K. Note that, considering the first 4 classes, the query melody was successfully identified more than a 97% of the times with $K = 3$. Therefore, we can say that the correct song usually was among the first solutions proposed by the system.

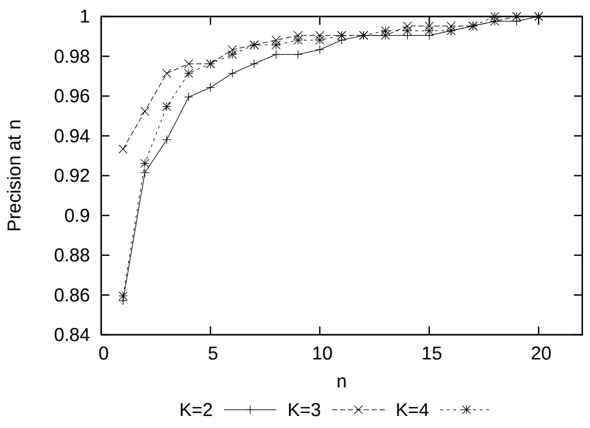


Fig. 2: Success rates as a function of the number of classes (songs) retrieved as the most probable ones.

5 Conclusions

In this paper, stochastic k-testable tree-models, formerly used in other domains like structured document compression or natural language processing, have been applied for computing the similarity between two melodies represented by trees.

This similarity is given as the probability of a song to belong to a class made up of different variations of that song as they were performed by a number of players.

Our goal was to identify a melody from a set of different variations and, in order to evaluate the proposed method, its performance has been compared to the same task and data but using a classical tree edit distance.

The results improved those obtained using the tree edit distance, showing that this probabilistic models are suitable for the classification of tree-represented music data. The high degree of variations may lead the edit distance to wrong decisions, but this probabilistic models deal better with noisy data.

This is an initial attempt, so we are persuaded that these promising results can be improved by adjusting discount parameters of the back-off model or using different and more sophisticated discount methods. Also other music categorization problems like genre or style classification will be explored.

Acknowledgements

This work is supported by the Spanish Ministry projects: DPI2006-15542-C04, TIN2006-14932-C02, both partially supported by EU ERDF, and the *Consolider Ingenio 2010* research programme (project MIPRCV, CSD2007-00018).

References

1. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18** (1989) 1245–1262
2. Selkow, S.M.: The tree-to-tree editing problem. Inf. Process. Lett. **6** (1977) 184–186
3. Rico-Juan, J.R., Calera-Rubio, J., Carrasco, R.C.: Smoothing and compression with stochastic k-testable tree languages. Pattern Recognition **38** (2005) 1420–1430
4. Illescas, P.R., Rizo, D., Iñesta, J.M.: Harmonic, melodic, and functional automatic analysis. In: Proc. of the 2007 International Computer Music Conference. Volume I. (2007) 165–168
5. Downie, J.S.: Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic n-grams as Text. PhD thesis, University of Western Ontario (1999)
6. Doraisamy, S., Rüger, S.M.: A polyphonic music retrieval system using n-grams. In: Proc. of ISMIR. (2004)
7. García, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 920–925
8. Knuutila, T.: Inference of k-testable tree languages. In: Bunke (Ed.), Proccedings of the International Workshop on Structural and Syntactic Pattern Recognition. (1993) 109–120
9. Ney, H., Essen, U., Kneser, R.: On the estimation of small probabilities by leaving-one-out. IEEE Trans. Pattern Anal. Mach. Intell. **17** (1995) 1202–1212
10. Habrard, A., Iñesta, J.M., Rizo, D., Sebban, M.: Melody recognition with learned edit distances. LNCS **5342** (2008) 86–96