Tree language automata for melody recognition

José F. Bernabeu, Jorge Calera-Rubio, José M. Iñesta, David Rizo
University of Alicante
{jfbernabeu, calera, inesta, drizo}@dlsi.ua.es

Abstract

The representation of symbolic music by means of trees has shown to be suitable in melodic similarity computation. In order to compare trees, different tree edit distances have been previously used, being their complexity a main drawback. In this paper, the application of stochastic k-testable tree-models for computing the similarity between two melodies as a probability, compared to the classical edit distance has been addressed. The results show that k-testable tree-models seem to be adequate for the task, since they outperform other reference methods in both recognition rate and efficiency. The case study in this paper is to identify a snippet query among a set of songs. For it, the utilized method must be able to deal with inexact queries and efficiency for scalability issues.

1 Introduction

Music pieces can be represented by symbolic structures such as strings or trees containing the sequence of notes in the melody. In linear representations, both melodic dimensions: time (duration) and pitch 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. Therefore, they are less sensitive to melody coding issues, since only pitch codes are needed to be established, so there are less degrees of freedom for coding.

A central problem in music information retrieval is to recognise copies/versions of a same song, although the versions may considerably differ from each other. An equivalent task is to recognize songs in a catalogue from (usually inexact) queries. In this paper, the problem of comparing symbolically encoded (e.g. MIDI) musical works is addressed. For it, probabilistic k-testable tree models [11], a generalization of the k-gram models, are utilized. They 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. These models have been used in a number of applications, like structured data compression [11] or information extraction from structured documents (like HTML or XML, for example) [8], but here they are applied to music data, focusing on its inherent structured nature. A preliminary version of the method proposed in this paper was presented in [2].

In the literature, several methods have been developed for comparing monophonic musical works using both edit [9] or geometric [1] distances. Recently, Habrard and coworkers [5] have introduced the approach of using probability as a way to estimate how similar two melodies are. In the present work, we will compare our results with theirs as a reference method.

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): $\Sigma_p = \{p \in \mathcal{N} \mid 0 \le p \le 11\} \cup \{`-'\}$. This way, in 'G Major', any pitch 'G' is mapped to 0. This alphabet permits a transposition invariant representation and keeps cardinality low. Rests are represented by a special label '–'.

In the proposed approach, each melody bar is represented by a tree, $t \in T_\Sigma$. Bars are coded by separated trees and then they are linked to a common root. Node labels are intervals from the tonic. The level of a node in the tree determines the duration it represents (see an example in Fig. 1). 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, nodes in level i represent duration of a $1/2^{i-1}$ of a bar. 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. In that moment, the corresponding leaf node is labeled with the pitch representation symbol, $\sigma \in \Sigma_p$.

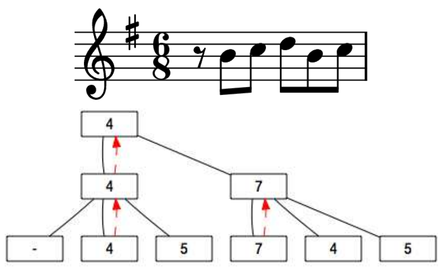

Figure 1: Tree representation of a one-bar melody.

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 [6]. 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. Notes have always higher priority than rests. Eventually all the internal nodes are labeled. The root of the forest is labeled with the root of the first tree.

3 Stochastic k-testable tree models

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 in some music tasks [3]. k-gram models can be regarded as a probabilistic extension of locally testable languages [13]. Informally, 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, together with prefixes and sufixes of length strictly smaller than k to check near the string boundaries. These models are easy to learn and can be efficiently processed.

Locally testable languages, in the case of trees, as described by Knuutila [7]. 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 descendants 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. These concepts are illustrated in Fig. 2 for the tree $t = a(a(a(ab))b)$. In this example, $r_2(t) = \{a(ab)\}$, $f_3(t) = \{a(a(a)b), a(a(ab))\}$, and $s_2(t) = \{a(ab), a, b\}$.

These kind of probabilistic tree languages can be defined using the formalism of deterministic tree automata (DTA), $A = (Q, \Sigma, \Delta, F)$. The procedure to infer this kind of automata from a training sample, $\Omega = \{t_1, t_2 ..., t_{|\Omega|}\}$, can be done easily [11]. For this pourpose, the parameters are computed as follows. The set of states is built as

$$Q = r_{k-1}(f_k(\Omega) \cup s_{k-1}(\Omega)) \; ; \qquad (1)$$

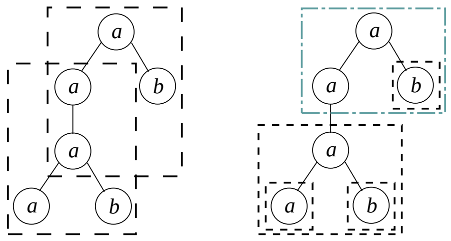

Figure 2: Left: set of 3-forks in $a(a(a(ab))b)$. Right: 2-root (in grey) and 2-subtrees (black dashed).

the alphabet Σ, is the set of pitch labels Σ_p in our case; the set of accepting states is

$$F = r_{k-1}(\Omega) \; ; \tag{2}$$

and Δ contains the transitions

$$\delta_m(\sigma, t_1, ..., t_m) = \begin{cases} r_{k-1}(\sigma(t_1 ... t_m)) & \text{if } \sigma(t_1 ... t_m) \in f_k(\Omega) \cup s_{k-1}(\Omega) \\ \perp & \text{otherwise} \end{cases} \tag{3}$$

where $\perp$ is an absorption state.

For instance, using the single sample $\Omega = \{a(a(a(ab))b)\}$ and for $k = 3$, one gets the set of states $Q = \{a(ab), a(a), a, b\}$ with final subset $F = \{a(ab)\}$ and transitions $\delta_0(a) = a$, $\delta_0(b) = b$, $\delta_2(a, a, b) = a(ab)$, $\delta_2(a, a(a), b) = a(ab)$ and $\delta_1(a, a(ab)) = a(a)$.

This learning procedure can be extended to the case where the sample Ω is stochastically generated incorporating probabilities to the DTA.

As shown in [11], a probabilistic DTA (PDTA) incorporates a probability, $p_m(\sigma, t_1, ..., t_m)$, for every transition in the automaton, with the normalization that the probabilities of the transitions leading to the same state $q \in Q$ must add up to one. For this purpose, one should note that, in this kind of deterministic models, the likelihood of training sample is maximized if the stochastic model assigns to every tree t in the sample a probability equal to its relative frequency in Ω [10]. So, these probabilities must be calculated as the ratio between the number of occurrences of a transition and the number of occurrences of the state to which this transition leads. PDTA also incorporate a probability $\rho(q)$ for every accepting state, $q \in F$. These probabilities are calculated as the ratio between the number of occurrences of a accepting state and the number of trees in the sample, $|\Omega|$. It is useful to store the above probabilities as the quotient of two terms. This way, if a new tree (or subtree) t is provided, the automaton can be easily updated to account for the additional information. For this incremental update, it suffices to increment each term with the partial sums obtained for t. Finally, the probability of the tree t is computed as $p(t) = \rho(\delta(t))\pi(t)$, where the product of the probabilities of all the transitions is recursively computed as:

$$\pi(t) = \begin{cases} p_0(\sigma) & \text{if } t = \sigma \in \Sigma \; , \\ p_m(\sigma, \delta(t_1), ..., \delta(t_m))\pi(t_1) \cdots \pi(t_m) & \\ & \text{if } t = \sigma(t_1 ... t_m) \in T_\Sigma - \Sigma \; . \end{cases} \tag{4}$$

Smoothing and classification

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 PDTA for each class, C_j, from well classified melodies.

Once the PDTAs for the different classes have been inferred and the probabilities estimated (see [11] for the details), a melody M is classified in the class $\hat{C}$ that maximizes the likelihood

$$\hat{C} = \arg\max_j \; l(M|C_j) \tag{5}$$

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

$$l(M|C_j) = \rho^{[2]}(\sigma|C_j) \prod_{i=1}^{|M|} \pi^{[k]}(t_i|C_j) \tag{6}$$

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 the common root is utilized, 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 [10]. 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 (K is the more specific model). 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, that is, with the more specific 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.

4 Results

In our experiments, we try to identify a problem melody using a set of different variations played by musicians for training. For that, we use a corpus consisting of a set of 420 monophonic 8-12 bar incipits 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 (to identify the part of the tune) 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.

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

The results using the proposed maximum likelihood technique were compared to those obtained for the same data and conditions tested in [5], where for each target melody, classical tree edit distances [12] to all the prototypes were computed and the 1-NN rule was applied. The edit distance was computed with insertion, deletion and substitution costs set to 1.

A 3-fold cross-validation scheme was carried out to perform the experiments, obtaining average success rates and dispersions (see Table. 1). Values for $K = \{2, 3, 4\}$ were utilized. Also, the values for the discount parameters where: for the forest root, $\theta^{[2]}(\sigma) = 5 \cdot 10^{-7}$ for all $\sigma \in \Sigma_p$, for the states, $\lambda_\sigma^{[1]}(m) = 0.001$ for all the arities, and for the rules, $\lambda^{[k]}(t) = 0.35$ for all the models and trees.

Table 1: Average success rates and comparison with tree edit distance.

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

Note that stochastic k-testable methods for all K significantly improve the results of the classical tree edit distance.

Fig. 3 shows the success rates taking into account the first n most probable classes for different values of K. Note that, with the first 5 classes, the success rate reaches around 97% with $K=3$. Therefore, we can say that the correct song usually was among the first solutions.

Another important point here is efficiency and scalability of the approach. It is remarkable that the classical tree edit distance has a high complexity, $O(m_A m_B \max\{h_A, h_B\})$, where m_i are the maximum arities of the two trees, and h_i their heights, which in practice is actually $O(|M_A||M_B| \max\{h_A, h_B\})$. This must be multiplied by the complexity (quadratic) of the nearest neighbour method implemented to perform the classification,

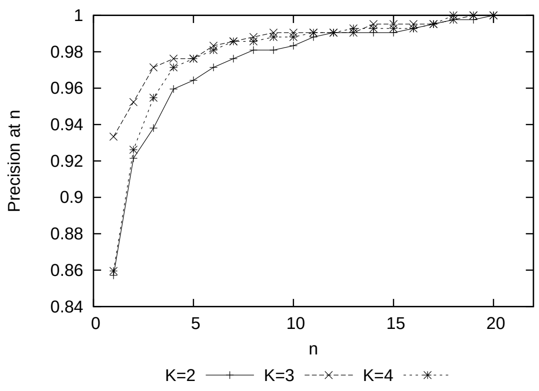

Figure 3: Success rates as a function of the number of classes retrieved as the most probable ones.

so if $n = |\Sigma|$ the total complexity is $O(n^2|M_A||M_B|\max\{h_A, h_B\})$. On the other hand, the proposed method has been shown [4] to run in $O(|M|^{k-1} n \log n)$ time and the classification is performed in linear time with $|t|$, once k is fixed.

5 Conclusions

In this paper, we applied stochastic k-testable tree-models showing that they are suitable for the classification of tree-represented music data. Our goal was to identify a melody from a set of different variations. The results overcame those previously obtained using the tree edit distance for the same corpus. We are persuaded that these promising results can be improved by adjusting some parameters not studied yet, like 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 project TIN2009-14247-C02-02 and the program CONSOLIDER INGENIO 2010 (CSD2007-00018).

References

[1] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, Y. Nuñez, D. Rappaport, and G. Toussaint. Algorithms for computing geometric measures of melodic similarity. *Computer Music Journal*, 30(3):67–76, Fall 2006.

[2] J. Bernabeu, J. Calera-Rubio, J. Iñesta, and D. Rizo. A probabilistic approach to melodic similarity. In *Proceedings of MML 2009*, pages 48–53, 2009.

[3] J. S. Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic n-grams as Text.* PhD thesis, University of Western Ontario, 1999.

[4] P. García. Learning k-testable tree sets from positive data. Technical Report DSIC/II/46/1993, Universidad Politecnica de Valencia, 1993.

[5] A. Habrard, J. M. Iñesta, D. Rizo, and M. Sebban. Melody recognition with learned edit distances. *LNCS*, 5342:86–96, 2008.

[6] P. R. Illescas, D. Rizo, and J. M. Iñesta. Harmonic, melodic, and functional automatic analysis. In *Proc. of the 2007 International Computer Music Conference*, volume I, pages 165–168, 2007.

[7] T. Knuutila. Inference of k-testable tree languages. In *Bunke (Ed.), Advances in Structural and Syntactic Pattern Recognition (Proc. of the S+SSPR'92), World Scientific, Singapore*, 1993.

[8] R. Kosalaa, H. Blockeela, M. Bruynooghea, and J. V. den Bussche. Information extraction from structured documents using k-testable tree automaton inference. *Data & Knowledge Engineering*, 58(2):129–158, August 2006.

[9] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[10] H. Ney, U. Essen, and R. Kneser. On the estimation of small probabilities by leaving-one-out. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(12):1202–1212, 1995.

[11] J. R. Rico-Juan, J. Calera-Rubio, and R. C. Carrasco. Smoothing and compression with stochastic k-testable tree languages. *Pattern Recognition*, 38(9):1420–1430, 2005.

[12] S. M. Selkow. The tree-to-tree editing problem. *Inf. Process. Lett.*, 6(6):184–186, 1977.

[13] Y. Zalcstein. Locally testable languages. *J. Comput. Syst. Sci.*, 6(2):151–167, 1972.