# Classifying melodies using tree grammars

José F. Bernabeu, Jorge Calera-Rubio, and José M. Iñesta

Dept. Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain
{jfbernabeu,calera,inesta}@dlsi.ua.es

**Abstract.** Similarity computation is a difficult issue in music information retrieval, because it tries to emulate the special ability that humans show for pattern recognition in general, and particularly in the presence of noisy data. A number of works have addressed the problem of what is the best representation for symbolic music in this context. The tree representation, using rhythm for defining the tree structure and pitch information for leaf and node labeling has proven to be effective in melodic similarity computation. In this paper we propose a solution when we have melodies represented by trees for the training but the duration information is not available for the input data. For that, we infer a probabilistic context-free grammar using the information in the trees (duration and pitch) and classify new melodies represented by strings using only the pitch. The case study in this paper is to identify a snippet query among a set of songs stored in symbolic format. For it, the utilized method must be able to deal with inexact queries and efficient for scalability issues.

**Keywords:** Music Modeling & Analysis, Stochastic Methods, Learning with Structured Data, Music Similarity, Classification.

## 1 Introduction

One of the main concerns in music information retrieval (MIR) tasks is how to assess melodic similarity in a similar way to how humans do. We are very good at recognizing previously known patterns, even if our perceived inputs are distorted, they are presented just partially, or in the presence of noisy data. This happens in music comparison in a number of situations, for example, when comparing different cover versions of a given melody or when searching in databases using a query that will be, by its own nature, partial and can be distorted or even wrong. Two issues are concerned to this problem: the similarity computation and the representation structure.

In this paper, the problem of comparing symbolically encoded (any format of digital scores) musical works is addressed. For it, we use probabilistic tree grammars [12]. These grammars can be obtained from probabilistic $k$-testable tree models [9]. The duration information implicit in the tree representation is captured by the grammar and this is used for classifying the new melodies represented by strings that only have the pitch information. This is a solution when duration information is not available or unreliable for the input data.

The results are compared with those obtained by Bernabeu [1] where input and training data are trees.

## 2 Melody tree representation

For representing the note pitches in a monophonic melody $s$ as a string, symbols $\sigma$ from a pitch representation alphabet $\Sigma_p$ 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 and the symbol '$-$' to represent rests. Thus the alphabet used is: $\Sigma_p = \{p \in \mathcal{N} \mid 0 \leq p \leq 11\} \cup \{\text{'}-\text{'}\}$. This way, in 'G Major', any pitch 'G' is mapped to 0 and the other pitches are represented by the number of semitones mod 12 from 'G'. This alphabet permits a transposition invariant representation and keeps cardinality low ($|\Sigma_p| = 13$).

In the tree approach, each melody bar is represented by a tree, $t \in T_{\Sigma_p}$. Bars are coded by separated trees and then they are linked to a common root. The level of a node in the tree determines the duration it represents. Each tree root (level 1) represents the duration of the whole bar. For a binary meter, the two nodes in level 2 represent 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 for a binary meters ($1/3^{i-1}$ for ternary). 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, $\sigma \in \Sigma_p$ (see [10] for details).

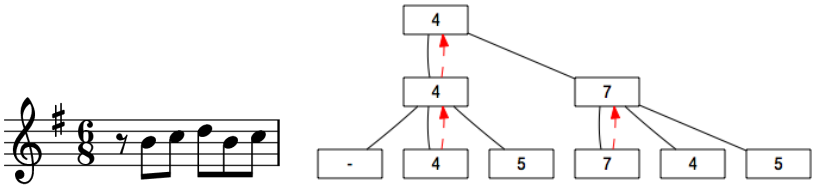



Fig. 1: Tree representation of a one-bar melody in a ternary meter with an example of how pith labels are propagated.

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 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 (Fig. 1 shows an example). Eventually,

when all the internal nodes are labeled, all bar trees are linked to a common forest root, labeled with the root of the first bar 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 [4]. $k$-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 suffixes 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, were 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 kind of probabilistic tree languages can be defined using the formalism of deterministic tree automata (DTA). The procedure to infer this kind of automata from a training sample, can be done easily (see [9] for details). This learning procedure can be extended to the case where the sample $\Omega$ is stochastically generated, incorporating probabilities to the DTA.

As shown in [9], 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 the training sample is maximized if the stochastic model assigns to every tree $t$ in the sample a probability equal to its relative frequency in $\Omega$ [8]. So, these probabilities must be calculated as the ratio of the number of occurrences of a transition to the number of occurrences of the state to which this transition leads. PDTA also incorporate a probability $\rho(q)$ for every accepting state, $q \in F$ ($F \subseteq Q$). These probabilities are calculated as the ratio between the number of occurrences of an 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 parsing $t$. Finally, the probability of the tree $t$ is computed as the product of the transitions utilized in the parsing of the tree (see [9] for details).

## 4 Grammars

At this point, we can classify a new melody in a particular class. For this purpose, 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 [9] for details), a melody $M$ can be classified in the class $\hat{C}$ that maximizes the likelihood (see [1]).

In order to do this, both training and new melodies must be represented by trees. However, what happens if the new melodies are only represented by strings? Moreover, what happens if a new melody string has the pitches but the duration information is not available? This situation appears when a melody query is given using only note pitches or when durations are not reliable. In other words, we have a set of melodies represented by trees to train the system but the target data are melodies represented by pitch strings. Therefore, we need to transform the $k$-testable tree automata in context-free grammars [12] in order to use them for parsing the input melody strings.

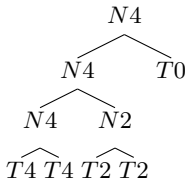
Context-free grammars may be considered to be the customary way of representing syntactical structure in natural language sentences. In many natural language processing applications, to obtain the correct syntactical structure for a sentence is an important intermediate step before assigning an interpretation to it. In our case, we use these grammars to obtain the correct structure for a given melody represented by a string.

Treebank grammars - which are explained in detail in [12]- are probabilistic context-free grammars in which the probability that a particular nonterminal is expanded according to a given rule is estimated as the relative frequency of that expansion by simply counting the number of times it appears in a manually-parsed corpus.

Before transforming our $k$-testable tree automata in context-free grammars we need introduce some changes in the melody tree representation. These changes are necessary because if we use the alphabet described in section 2 then a transition in the automaton could be transformed in different grammar rules. This happens because we can not distinguish between symbols that are terminals or nonterminals. In order to solve these ambiguities we need to label tree nodes adding the symbol '$T$' to that of $\Sigma_p$ if it is a leaf node (terminal) and the symbol '$N$' if it is an inner node (nonterminal).

Therefore, if $\Omega = t_1, t_2, \ldots, t_{|\Omega|}$ is a treebank, that is, a stochastic sample of parse trees, the alphabet $\Sigma$ can be safely partitioned into the subset $s_1(\hat{\Omega})$ of labels that may only appear at leaves ($\Sigma_{pT} =$ '$T$'$\Sigma_p$) and its complementary subset $\Sigma - s_1(\hat{\Omega})$ ($\Sigma_{pN} =$'$N$'$\Sigma_p$) of labels at internal nodes (propagated labels in the trees).

Then, we can define a probabilistic $k$-testable grammar as $G^{[k]} = (V^{[k]}, T, I, R^{[k]}, p^{[k]})$, where $V^{[k]} = I \cup r_{k-1}(f_k(\Omega) \cup s_{k-1}(\hat{\Omega})) - s_1(\hat{\Omega})$ is the set of nonterminals, $T = s_1(\hat{\Omega})$ is the set of terminals, $I$ is the start symbol, $R^{[k]}$ is the set of production rules, and $p^{[k]}$ a probability function (see [12] for details).




(1/1) $I \to N4$
(1/3) $N4 \to N4\ \ T0$
(1/3) $N4 \to N4\ \ N2$
(1/3) $N4 \to T4\ \ T4$
(1/1) $N2 \to T2\ \ T2$

Fig. 2: Example of a probabilistic tree grammar for $k = 2$.

Figure 2 shows the corresponding probabilistic context-free grammars (PCFG) (right) according to the tree in left. For this tree we have the sets (roots) $r_1(t) = N4$, (forks) $f_2(t) = \{N4(N4\,T0), N4(N4\,N2), N4(T4\,T4), N2(T2\,T2)\}$, (subtrees) $s_1(t) = \{T0, T2, T4\}$. Therefore, we can obtain $k$-testable grammars with different values for $k$.

### 4.1 Smoothing

In general, $k$-testable grammars with larger values of $k$ contain more specialized rules and, therefore, are less ambiguous and allow for faster parsing. In contrast, typical treebank grammars have 100 percent coverage (as remarked in [3]) unlike with higher order grammars where sentences without a valid parse tree are common. Therefore, the use of smoothing techniques becomes necessary if one wants to use these models for parsing. Two classical techniques of this type are linear interpolation and backing-off [8].

Smoothing through linear interpolation is performed by computing the probability of events as a weighted average of the probabilities given by different models. This approach has a problem when the higher order models return a zero probability due to unseen labels. Then, a new melody is classified only with the more general and more ambiguous model discarding the entire more specific model.

In contrast, backing-off allows to avoid lower-order parsing when possible. In other words, backing-off tries to parse with the higher-order grammar unless no parse tree is provided by this grammar. Only in such a case the lower-order model is called, so backing-off is faster than linear interpolation. However, the lack of a single rule in the sample can force the parser to use the lower-order model, loosing all the higher-order information for a whole sentence.

Here, we use an alternative approach: the rule-based backing-off [12]. Using this rule-based backing-off requires the implementation of specific parsers since building the whole grammar is unfeasible due to the large number of implicit rules (even if only those assigning a strictly positive probability in the last case of [5] are considered). An alternative scheme that requires only minor modifications is to use a quasi-equivalent grammar $G'$ built as in [12].

However, we need to define a universal grammar because some labels in $\Sigma_p$ for the leaves of the trees do not appear in the training data for the grammars. Then, if a particular new melody contains an unseen label, the parser will return

a zero probability. For solving this problem we only have to introduce the rules of the form $N\sigma_1 \to T\sigma_2$ (where $\sigma_1, \sigma_2 \in \Sigma_p$) (updating the corresponding counters) if the rule did not appear during training. Introducing these rules the grammar assigns a non zero probability for each string even if the label does not appear in the training data.

### 4.2 Classification

As explained before, we want to study if the proposed approach can be used to classify new melodies represented by strings. After a grammar $G_j$ is inferred for each class $C_j$ we need an algorithm for obtaining the probability that a given string $s$ is generated by a grammar $G_j$. For this purpose, we have used the Stolcke algorithm [11] and the CYK+ algorithm [2] for string parsing. These parsing algorithms are able to give the probability $p(s|G)$ that a string $s$ is generated by a probabilistic Context-free grammar $G$ without requiring conversions to Chomsky Normal Form (CNF). Then, a melody $M$ is classified in the class $\hat{C}$ that maximizes the likelihood

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

We can calculate this likelihood two ways: splitting the melody (SplitBars) in bars or computing the whole melody (Whole).

In SplitBars, the melody string is split in $|M|$ (number of bars in a melody $M$) bar strings $s_1, \ldots, s_{|M|}$. Therefore we are able to compute the probability of each bar string to belong to a particular class (grammar). Suppose we have a finite number of classes and we have computed the membership probability of each bar string $s_i$ to each of these grammars, $p(s_i|G_j)$. These probabilities can be combined to give a decision for the whole song. For the combination of bars the geometric mean has been used. The geometric mean is less sensitive to outliers than the arithmetic one. For our purposes is enough to multiply all bar strings probabilities of the whole melody. Calculating the $|M|$-th root of the resulting product is not needed for classification because, given a particular melody $M$ to classify, $|M|$ is the same for all classes. Therefore,

$$l(M|C_j) = \prod_{i=1}^{|M|} p(s_i|G_j) \tag{2}$$

On the other hand, in the Whole strategy we only need the probability of the melody string (now $M = s$). Then

$$l(M|C_j) = p(s|G_j) \tag{3}$$

For calculating this probability we need to introduce the start rules $S \to IS$ and $S \to I$ which define a melody recursively (melody is formed by a bar and a melody) ($I$ is the initial symbol for a bar as explain in section 4). Therefore, the grammars allow to recognize a whole melody instead of melodies split in bars.

## 5 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.

A 3-fold cross-validation scheme was carried out to perform the experiments, obtaining average success rates and dispersions ($(max - min)/4$).

Table 1: Success rates with the different approaches used.

| Approach | Success rate |
|---|---|
| PDTA | $87.3 \pm 0.7$ |
| StringBars | $92.4 \pm 1.1$ |
| Whole | $86.9 \pm 1.3$ |

Table 1 shows the results of classification using the approaches explain in section 4.2. These results are compared with the results using the approach of probabilistic deterministic tree automata used in [1] but using the notation change described in section 4.

Note that PDTA uses the duration information implicit in tree representation, however the grammar approaches use less information (only pitch) for classifying. From the results, it is observed that the new approach using the strings through the StringsBars method improves significantly the PDTA results. The Whole approach did not improve the results because it is more sensitive to variations in the data than the geometric mean of the bar probabilities.

## 6 Conclusions

In this paper, we applied probabilistic tree grammars constructed from stochastic $k$-testable tree-models showing that this approach can be used for classifying new melodies represented by strings using the information captured in the grammar rules. This approach allows avoiding the duration information in the input data (strings with pitch only), making easier querying a music database. Our goal was to identify a melody from a set of different variations. The results overcame those

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

previously obtained using probabilistic deterministic tree automata for the same corpus. According to the results, we can say that the classification is improved splitting the melody in bars. Also the results keep in good performance taking the string of the whole melody, which is important since not always the bar information is available. We are persuaded that these promising results can be improved by defining a more complex universal grammar for unseen labels and removing some rules that make the grammars more ambiguous.

**Acknowledgements** This work is supported by the Spanish Ministry project TIN2009-14247-C02-02, TIN2009-14205-C04-C1, and the program Consolider Ingenio 2010 (CSD2007-00018).

## References

1. J. F. Bernabeu, J. Calera-Rubio, J. M. Iñesta, and D. Rizo. A probabilistic approach to melodic similarity. In *Proceedings of MML 2009*, pages 48–53, 2009.
2. Jean-Cédric Chappelier and Martin Rajman. A generalized cyk algorithm for parsing stochastic cfg. In *TAPD*, pages 133–137, 1998.
3. Eugene Charniak. Tree-bank grammars. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036, 1996.
4. J. Stephen Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic n-grams as Text.* PhD thesis, University of Western Ontario, 1999.
5. Lyn Frazier and Keith Rayner. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14(2):178 – 210, 1982.
6. Plácido R. Illescas, David Rizo, and José 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. Hermann Ney, Ute Essen, and Reinhard Kneser. On the estimation of small probabilities by leaving-one-out. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(12):1202–1212, 1995.
9. 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.
10. D. Rizo, K. Lemström, and J. M. Iñesta. Tree representation in combined polyphonic music comparison. *Computer Music Modeling and Retrieval. Genesis of Meaning in Sound and Music. Lecture Notes in Computer Science*, 5493:177–195, 2009.
11. Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21:165–201, 1995.
12. J. L. Verdu-Mas, R. C. Carrasco, and J. Calera-Rubio. Parsing with probabilistic strictly locally testable tree languages. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1040–1050, 2005.
13. Yechezkel Zalcstein. Locally testable languages. *J. Comput. Syst. Sci.*, 6(2):151–167, 1972.