MIREX 2015 SYMBOLIC MELODIC SIMILARITY: APPLYING STRING SIMILARITY MEASURES TO PITCH/REST SEQUENCES

Shiho Sugimoto, Yuto Nakashima, Masayuki Takeda
Department of Informatics, Kyushu University, Japan
{shiho.sugimoto, yuto.nakashima, takeda}@inf.kyushu-u.ac.jp

ABSTRACT

In this report, we describe several similarity measures which we use for the task of Symbolic Melodic Similarity (SMS) in MIREX 2015.

1. INTRODUCTION

We consider similarity measures between music pieces s that are combinations of a preprocessing p of music pieces into particular sequential representation, and a string similarity measure δ. That is, d is given by $d(x, y) = \delta(p(x), p(y))$, where x, y are music pieces.

In the research field of string algorithms many string (dis)similarity measures have been proposed, such as the edit distance [7], the longest common subsequence (LCS) length [2], the normalized compression distance (NCD) [1], which are used in many applications such as automatic spelling correction, information retrieval, gene information analysis and so on. String kernels such as the n-gram kernel [5], the mismatch kernel [4], and the subsequence kernel [6], which are used in the machine learning field, are also string similarity measures.

2. OVERVIEW

From MIDI files, we remove MIDI events other than the NOTE ON/OFF events and quantized the NOTE ON/OFF times with unit time corresponding to the sixteenth note length. We merge all the tracks/channels of each MIDI file into one. The result can then be viewed as sequences of pitch sets that are "ON" in respective unit time intervals. Since the input music pieces are assumed to be monophonic, the pitch sets are empty or singleton. By ignoring octave differences, we can transform input music pieces into strings consisting of 13 symbols representing pitches and rests.

Let x, y be pitch/rest sequences obtained from MIDI files. Let $x \oplus d$ denotes the pitch/rest sequence obtained from x by transposing the key of x by d semi-tones. Then the similarity between x and y is defined by

$$SIM_s(x, y) = \max\{s(x \oplus d, y) \mid d = 0, 1, \ldots, 11\},$$

where s is a similarity measure on strings.

3. NCD WITH LZF

The Normalized Information Distance (NID) between two strings x and y is defined as follows.

$$NID(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}},$$

where $K(\cdot)$ and $K(\cdot|\cdot)$ denote the Kolmogorov complexity and the conditional Kolmogorov complexity, respectively. Letting $K(x|y) \approx K(yx) - K(y)$ and replacing the Kolmogorov complexity $K(\cdot)$ by compressed data size $C(\cdot)$ with an appropriate compression program C, we obtain the Normalized Compression Distance (NCD). More formally, the NCD between strings x and y is then defined as follows.

$$NCD_C(x, y) = \frac{\max\{C(xy) - C(x), C(yx) - C(y)\}}{\max\{C(x), C(y)\}},$$

where $C(z)$ denotes the compressed size of string z using a compression program C. It should be stated that we do not assume $C(yx) = C(xy)$ in the above equation in order to keep $NCD_C(x, y) = NCD_C(y, x)$.

The NCD values strongly depend on the compressor C. We use the Lempel-Ziv factorization (LZF), which is an abstraction of LZ77 compression algorithm.

Definition 1 (LZ-factorization) *The LZ-factorization of a string T is the factorization $T = f_1 \cdots f_n$ where each LZ-factor $f_k \in \Sigma^+$ $(k = 1, \ldots, n)$ is defined inductively as follows: $f_1 = T[1]$. For $k \geq 2$: if $T[|f_1 \cdots f_{k-1}| + 1] = c \in \Sigma$ does not occur in $f_1 \cdots f_{k-1}$, then $f_k = c$. Otherwise, f_k is the longest prefix of $f_k \cdots f_n$ that occurs at least twice in $f_1 \cdots f_k$.*

4. FLDC KERNEL

Let m and k be fixed integers with $m > 0$ and $0 \leq k \leq m$. Let

$$F_{(m,k)} = \{q \mid q \in (\Sigma \cup \{\bullet\})^m \text{ and } \#(q) \leq k\},$$

where $\#(q)$ denotes the number of occurrences of $\bullet$ within q. We note that when $k = 0$ the set $F_{(m,k)}$ is the set of strings of length m. The FLDC kernel is defined as follows.

Definition 2 (FLDC kernel) *For any strings $s, t \in \Sigma^*$, we define the* FLDC Kernel $K_f(x, y)$ *by*

$$K_f(x, y) = \sum_{q \in F_{(m,k)}} \mathit{Freq}_q(x) \cdot \mathit{Freq}_q(y).$$

We note that the FLDC kernel with $k = 0$ is the N-gram kernel. In this sense it is a natural extension of the N-gram kernel that allows up to k mismatches.

The FLDC kernel is closely related to the mismatch kernel in the sense that both the kernels allow mismatches up to k in strings of length m. The difference is in that the positions at which mismatches are allowed in strings are explicitly specified with $\bullet$'s in the FLDC kernel. Suppose that $(m, k) = (5, 2)$, and input strings s and t, respectively, contain $x = aaaac$ and $y = bbbbc$ as substrings. The strings x and y are dissimilar in the sense that their Hamming distance $d(x, y) = 4$ is greater than the threshold k. But we have $d(x, z) = d(y, z) = 2 \leq k$ for any $z \in \{aabbc, ababc, abbac, baabc, babac, bbaac\}$ and thus they increase the kernel value in the mismatch kernel. On the other hand, the strings x and y have no common FLDC patterns in $F_{(5,2)}$ and do not affect the kernel value in the FLDC kernel.

Although the feature space of the FLDC kernel is larger than that of the mismatch kernel, the time complexity of computing the kernel values for the FLDC kernel is the same as that for the mismatch kernel.

One naive method would be, for all patterns q in $F_{(m,k)}$, to compute the frequencies $\mathit{Freq}_q(x)$ and $\mathit{Freq}_q(y)$ of q in given strings x and y. The size of $F_{(m,k)}$ is, however:

$$|F_{(m,k)}| = \sum_{i=0}^{k} \left(\begin{array}{c} m \\ i \end{array} \right) \sigma^{m-i} = O(m^k \sigma^m),$$

where σ denotes the alphabet size. The total time is $O((|x|+|y|)m^{k+1}\sigma^m)$, and thus the method is impractical. An $O(mk)$ time pre-computation enables us to compute $K_f(x, y)$ in $O(m|x||y|)$ time for any strings s and t. By using a similar idea of the technique used for accelerating the mismatch kernel computation [3], a further improvement is possible.

Theorem 1 *For any strings x and y, $K_f(x, y)$ can be computed in $O(c_{m,k}(|x| + |y|))$ time, where $c_{m,k}$ is a constant independent of the alphabet size.*

5. REFERENCES

[1] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

[2] D.S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[3] Pavel P. Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Scalable algorithms for string kernels with inexact matching. In *Proc. 22nd Annual Conference on Neural Information Processing Systems (NIPS'08)*, pages 881–888, 2008.

[4] Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[5] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'02)*, pages 566–575, 2002.

[6] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, Chris Watkins, and Bernhard Scholkopf. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2002.

[7] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.