A New Tree Distance Metric for Structural Comparison of Sequences

Matthias Gallé

Symbiose Project
IRISA/INRIA Rennes-Bretagne Atlantique, France

Abstract. In this paper we consider structural comparison of sequences, that is, to compare sequences not by their content but by their structure. We focus on the case where this structure can be defined by a tree and propose a new tree distance metric that capture structural similarity. This metric satisfies non-negativity, identity, symmetry and the triangle inequality. We give algorithms to compute this metric and validate it by using it as a distance function for a clustering process of slightly modified copies of trees, outperforming an existing measure.

1 Motivation

Comparison of sequences is a recurrent task in computer science related applications, and had been a striking force in interdisciplinary fields like bioinformatics. Most of the existing comparison methods are based on the content and the sequential composition of these sequences. However, those measures do not apply well for comparing different sequences that may have a similar structure.

In this paper we consider the problem of comparing sequences by their structure rather than by their content. We suppose that the structure of a sequence is represented by an ordered tree whose leaves read from left to right yields the sequence. We do not consider how to obtain this structure, a great challenge itself [7], but on how to compare it, if it can be expressed in the form of a tree.

Our interest in comparing trees that represent the structure of a sequence is motivated from the use of straight-line context free grammars. This are special grammars that generate only one sequence. Several algorithms exists that, taken a string as input, tries to find a "good" grammar, where "good" is generally defined to be as smallest as possible (the problem of finding a grammar of minimal size that generates a given string is NP-Hard [6]). Interest in this algorithms comes from their compression capacity [8, 4] and their potential to be used in structure discovery [12]. However, results in structure discovery using this tools has been very limited and mostly picked up by eye. We think that this is partly due to that no meaningful measure exists that can be used in an automatic fashion to capture similarity.

Comparing trees is however a recurrent task in fields like natural language processing. One of the main problems in this area is, given a sequence of letters, words or part-of-speech tags, to automatically find the correct syntax tree. This is then compared to a manually annotated, "correct" tree, the *gold standard*. Despite of criticism [13], the Parseval metric [1], or an unlabeled variant of it [9, Section 2.2], seems still to be the standard metric. This measure seems to work for natural-language processing, where

Dagstuhl Seminar Proceedings 10231
Structure Discovery in Biology: Motifs, Networks & Phylogenies
http://drops.dagstuhl.de/opus/volltexte/2010/2737

the sentence are not very long, and a small change in the brackets suppose a huge change in the meaning. But it is not so clear how well this extends to bigger sequences, and when there are no clues about the underlying structure. The Parseval metric is based on an *exact* comparison between nodes, and is not flexible to contemplate matches due to *similar* nodes. We address this issue in our distance metric.

In this paper we present a distance metric for trees that captures their similarity if they are considered to be the structure of a sequence. It is an easy-to-implement metric which is flexible to small differences and a straightforward extension permits to compare trees of different size. This paper is structured as follows: Sect. 2 defines the notation we use, Sect. 3 is the central part of this paper, where we define our metric and prove that it is a proper distance function. In Section 4 we analyze algorithms to compute it and in Section 5 we test its ability to classify tree that were slightly modified.

2 Preliminaries

We index sequences starting from 1. Trees will be denoted by capital letters, a node is called *internal* if it has at least one daughter and *leaf* otherwise. The *yield* of tree T is the sequence composed by its leaves, read from left to right. Any node n defines a subtree T_n where n is the root. Clearly, *yield*(T_n) is a substring of *yield(T)*. We define the *interval* of a node n to be the interval corresponding to the positions of *yield*(T_n) in *yield*(T). Finally the *bracket set* of a tree T is the set of intervals given by the internal nodes (except the root) of T. Brackets will be denoted by lowercase letters. With the exception of the label of the nodes, the parse tree can be fully recovered from this set of intervals, so we will use as equivalent parse trees and bracket set.

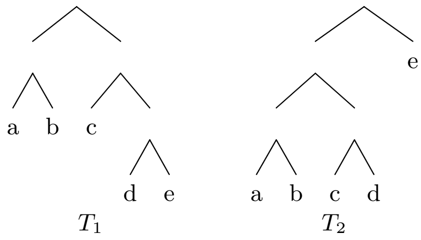

Fig. 1. Two different trees with the same yield.

Example Consider Fig. 1. The bracket set of tree T_1 is $\{[1,5],[1,2],[3,5],[4,5]\}$ and the one of tree T_2 is $\{[1,5],[1,4],[1,2],[2,3]\}$.

The unlabeled version of the Parseval metric used to evaluate syntax tree is the Unlabeled F-Measure (the harmonic mean between the precision and recall). It is equal to the Dice coefficient:

Definition 1 (Dice coefficient). *Given sets X, Y.*

$$Dice(X,Y) = 2 * \frac{|X \cap Y|}{|X| + |Y|}$$

In Fig 1, $Dice(T_1, T_2) = 0.5$

3 A Similarity Measure for Trees

In this paper we propose a similarity metric between trees that ignores the labels of the nodes. It is inspired by the unlabeled version of the Parseval metric, but our measure turns out to be much more robust to small changes of the brackets. The motivation itself is also different: while the goal of the Parseval metric is to evaluate how close a proposed tree comes to a gold standard, the objective of our measure is to compare different trees over possibly different sequences. It tries to extract all possible similarities, even if the matches of brackets are not perfect.

As a first step we focus on comparison between two brackets, which are represented as intervals. In order to measure how similar they are, we use the well-known Jaccard coefficient that gives a measure of similarity of two sets:

Definition 2 (Jaccard Coefficient). *Given sets x,y,*

$$J(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

Here, we suppose that the brackets are intervals over the integer line, so that each bracket defines a finite set of integers.

This gives a similarity coefficient between 0 and 1, 0 being completely different (empty intersection) and 1 total equality ($x = y$). The corresponding distance measure is $d(x, y) = 1 - J(x, y)$.

Lemma 1. *d is a proper metric, that is, it satisfies:* non-negativity *($d(x, y) \geq 0$),* identity *($d(x, x) = 0$),* symmetry *($d(x, y) = d(y, x)$) and* triangle inequality *($d(x, y) + d(y, z) \geq d(x, z)$)*

The first three are trivially true. For the triangle inequality see [11].

In order to be able to compare the brackets of two trees (set of brackets) X and Y, we suppose an assignment function $f : X \to Y \cup \{\emptyset\}$. Moreover, we require this function to be injective, except possibly for $\emptyset$. This means, every bracket y from Y has at most one x from X such that $f(x) = y$. Let $R(f)$ denote the range of function f. Note that $R(f)$ may be only a subset of Y. The role of the empty set in the image is to permit to assign brackets from X which otherwise would not be assigned. If $f(x) = \emptyset$ we refer to this as a *null-assignment*. Note that $d(x, \emptyset) = 1$, the maximal value for d.

We compare two set of brackets as follows:

Definition 3. $D_f(X, Y) = \sum_{x \in X} d\,(x, f(x)) + |Y \setminus R(f)|$

This gives a penalty of a maximal distance for every bracket of Y to which no bracket of X was assigned. This is the symmetric part of assigning $\emptyset$ to a bracket of X.

In order to find the shortest distance between two trees, we are interested in finding the "best" possible assignment function. This is the function $f^* = \arg\min_f D_f(X, Y)$. Then, we define:

Lemma 2 (Distance measure). $D(X, Y) = D_{f^*}(X, Y)$

3.1 Symmetry

In order to prove symmetry, we define f^{-1} the inverse of a function in a non-standard way. Let f be as defined before. Then:

Definition 4 (Inverse function). $f^{-1} : Y \to X \cup \{\emptyset\}$

$$f^{-1}(y) = \begin{cases} x \textit{ if } y \in R(f) \textit{ and } f(x) = y \\ \emptyset \textit{ if } y \notin R(f) \end{cases}$$

So, each bracket from the image that was not assigned by f, receives a null-assignment of f^{-1}.

Lemma 3. $D_f(X,Y) = D_{f^{-1}}(Y,X)$

Proof.

$$\begin{aligned} D_{f^{-1}}(Y,X) &= \sum_{y \in Y} d\left(y, f^{-1}(y)\right) + \left|X \setminus R\left(f^{-1}\right)\right| \\ &= \{\text{Definition 4}\} \\ &\quad \sum_{x \in X \cap R(f^{-1})} d(f(x),x) + \sum_{y \in Y \setminus R(f)} d(y,\emptyset) + |X \setminus R(f^{-1})| \\ &= \{d(y,\emptyset) = 1\} \\ &\quad \sum_{x \in X \cap R(f^{-1})} d(f(x),x) + |Y \setminus R(f)| + \sum_{x \in X \setminus R(f^{-1})} d(x,\emptyset) \\ &= \{\text{symmetry of } d\} \\ &\quad \sum_{x \in X} d\left(x, f(x)\right) + |Y \setminus R(f)| \end{aligned}$$

□

Now, if f does not assign brackets y from Y, the y is null-assigned by f^{-1}. So, if f^* minimizes $D_f(X,Y)$, then $(f^*)^{-1}$ minimizes $D_g(Y,X)$. We have as corollary:

Corollary 1 (Symmetry). $D(X,Y) = D(Y,X)$

3.2 Triangle Inequality

For the proof of the triangle inequality, we will proceed similarly as before, and redefine the composition of function.

Definition 5 (Composition). *If* $f : X \to Y \cup \{\emptyset\}$ *and* $g : Y \to Z \cup \{\emptyset\}$*, then* $g \circ f$ *denotes the function:*

$$\begin{aligned} g \circ f \quad &: X \to Z \cup \{\emptyset\} \\ (g \circ f)(x) &= \begin{cases} g(f(x)) & \textit{if } f(x) \neq \emptyset \\ \emptyset & \textit{otherwise} \end{cases} \end{aligned}$$

See Figure 2 for illustration. The intuition behind is that, if while going from X to Z over Y an x gets a null-assignment, then the final assignment is also null.

We will make use of the two following lemmas:

Lemma 4. $|Z \setminus R(g \circ f)| \leq |Z \setminus R(g)| + |Y \setminus R(f)|$

Proof. By Definition 5, $|Z \setminus R(g \circ f)| \leq |Z \setminus R(g)|$ □

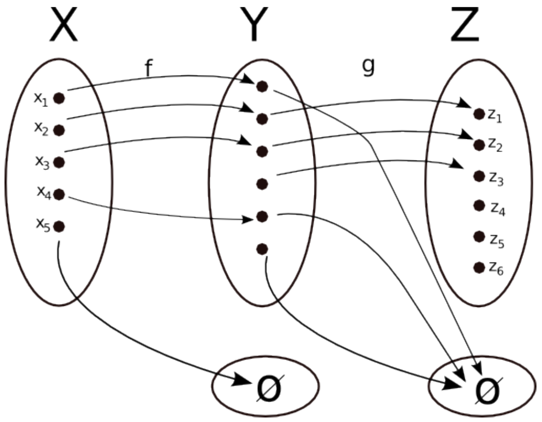

Fig. 2. In this case $(g \circ f)(x_2) = z_1, (g \circ f)(x_3) = z_2$ and the rest are null-assignments.

Lemma 5. $\displaystyle\sum_{x \in X} d(x, (g \circ f)(x)) \leq \sum_{y \in Y} d(y, g(y))$

Proof.

$$\begin{aligned}
&\sum_{x \in X} d\left(x, (g \circ f)(x)\right) \\
&= \sum_{x \in X, f(x) \neq \emptyset} d(x, (g \circ f)(x)) + \sum_{x \in X, f(x) = \emptyset} d(x, \emptyset) \\
&\leq \{\text{triangle inequality of } d\} \\
&= \sum_{x \in X, f(x) \neq \emptyset} \left(d\left(x, f(x)\right) + d\left(f(x), (g \circ f)(x)\right)\right) + \sum_{x \in X, f(x) = \emptyset} d(x, f(x)) \\
&= \{\text{Definition 5}\} \\
&= \sum_{x \in X, f(x) \neq \emptyset} d(x, f(x)) + \sum_{x \in X, f(x) = \emptyset} d(x, f(x)) + \sum_{x \in X, f(x) \neq \emptyset} d(f(x), g(f(x))) \\
&\sum_{x \in X} d(x, f(x)) + \sum_{y \in R(f) \setminus \{\emptyset\}} d(y, g(y)) \\
&\leq \sum_{x \in X} d(x, f(x)) + \sum_{y \in R(f) \setminus \{\emptyset\}} d(y, g(y)) + \sum_{y \in Y \setminus R(f)} d(y, g(y)) \\
&= \sum_{x \in X} d(x, f(x)) + \sum_{y \in Y} d(y, f(y)) \qquad \square
\end{aligned}$$

Corollary 2 (Triangle Inequality). $D(X, Y) + D(Y, Z) \geq D(X, Z)$

Proof. Suppose f, g, h are, respectively, the functions that minimize $D_e(X, Y), D_e(Y, Z)$ and $D_e(X, Z)$.

$$\begin{aligned} D(X,Z) &= D_h(X,Z) \\ &\leq \{\text{h minimizes } D_e(X,Z)\text{, so in particular } D_h(X,Z) \leq D_{g\circ f}(X,Z)\} \\ &\quad \sum_{x \in X} d(x,(g\circ f)(x)) + |Z \setminus R(g\circ f)| \\ &\leq \{\text{Lemma 5 and Lemma 4}\} \\ &\quad \sum_{x\in X} d(x,f(x)) + \sum_{y \in Y} d(y,f(y)) + |Z\setminus R(g)| + |Y \setminus R(f)| \\ &= D(X,Y) + D(Y,Z) \qquad \square \end{aligned}$$

4 Computation

In this section we consider an algorithm to compute the measure D and analyze its computational complexity.

The Jaccard coefficient can be computed in constant time because we restrict the sets to be intervals. Given the assignment function f, $D_f(X,Y)$ is computable in $\mathcal{O}(n)$, where $n = \max(|X|,|Y|)$. Note that we only consider trees where every node has at least two daughters, so the size of the set of brackets ($|X|$) is linear in the number of leaves (the length of the yield).

The computation of f^* can be mapped to optimize an assignment problem and can be solved, for example, by the Hungarian algorithm [10], which is in $\mathcal{O}(|X|^3)$. So, $D(X,Y)$ can be computed in $\mathcal{O}(n^3)$, where $n = \max(|X|,|Y|)$.

5 Experimentations

In order to test our distance measure and compare it to existing metrics we want to see if it is capable to distinguish groups of similar trees. For this, we start with a small set of radical different trees. We then modify copies of these original trees and use a clustering algorithm based on the distance metric to regroup them. The nature of the change is always the same, but is parametrized by a random distribution.

Starting from k sets of brackets, we copy each of them m times with some modifications. We use $k = 3$: a left-branching tree ($\{(i,n) : 1 \leq i < n-1\}$), a right-branching tree ($\{(0,i) : 1 < i < n\}$) and a centered tree ($\{(i, n-i) : 1 < i < \lfloor n/2 \rfloor\}$). In our set-up, $n = 30$ and we generate $m = 32$ modified copies of each one, resulting at the end in 99 bracket sets (we keep the 3 original trees). The modifications are obtained by changing each bracket with probability p. A change consists in a shift of the bracket to the left or to the right (choosing randomly). Shifting a bracket $[a,b]$ by ℓ consist in replacing it by $[a+\ell, b+\ell]$. The value ℓ is given by a geometric distribution with parameter q (plus one, to ensure that the bracket is changed). After each change, all overlapping brackets are shorten to avoid overlap. In order to not give preference to any bracket, the order in which they are considered is determined randomly. Throughout our experiments we use $q = 0.5$ and different values for p. For the clustering algorithm, we compute the square distance matrix and use a k-medoid algorithm ($k = 3$), taking the cluster with lowest total sum of distances to the center after 20 runs (each run starts with a random selection of centers).

Note that the Dice dissimilarity metric $(1 - D(X,Y)$ does not satisfy the triangle inequality and is inadequate to be used to compute a distance matrix. It is however closely related to the Tanimoto distance ($Dice(X,Y) = \frac{2*J(X,Y)}{1+J(X,Y)}$), so we compared our distance metric to the Tanimoto distance, but this time applied to set of brackets. In Fig. 3, we plot p (the probability that a bracket is changed) against the number of hits of the final cluster. Each point is the average over 250 runs, each run consisting in a copy-modify-cluster-count step. As it can be appreciated, as the probability of modification increases, the Tanimoto distance becomes less accurate to discriminate the correct clusters. This reveals the binary nature of a match in the Tanimoto distance: or a bracket matches or not. Our distance is more flexible: this reveals to be counterproductive if there are only few changes, but if p increases, it reveals to be very well suited to cluster the right groups. Both measures results in the same number of hits for $p = 0.5$, but from there on the number of hits using the Tanimoto distance decreases considerable. Our D distance continues to improve, getting more or less stable at 92 correct hits.

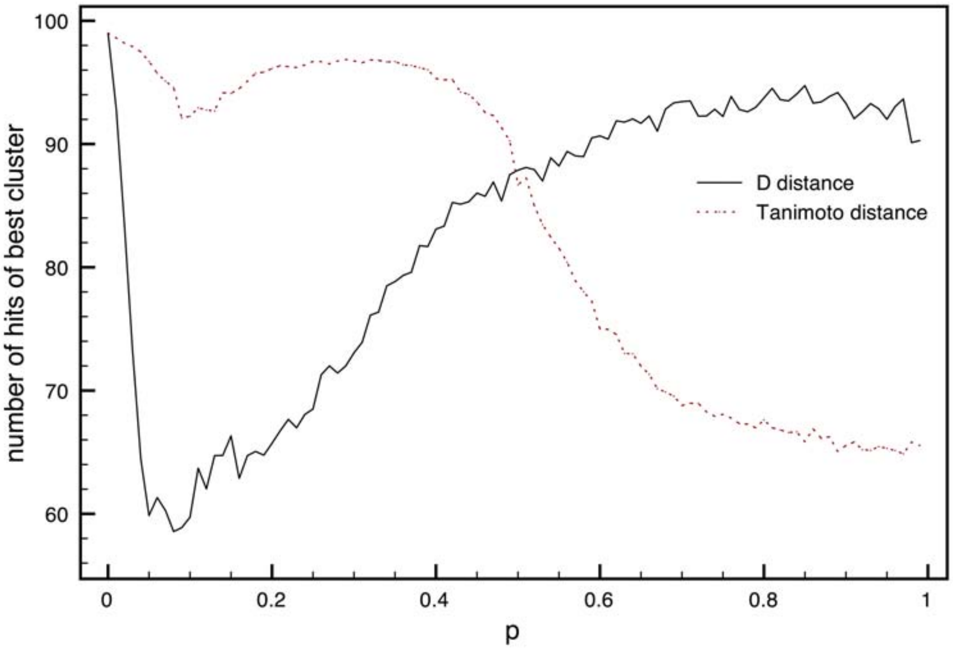

Fig. 3. Number of true positives of the final cluster against probability of change of a bracket.

6 Conclusions

We presented a new distance metric to compare trees and prove that it is a proper metric. The aim of this metric is to compare sequences based on their tree structure. The advantage over previous approaches is its flexibility to compare trees that intuitively are highly similar, but where existing similarity metrics fail.

Our experiments show that this metric permits to distinguish groups of similar parse tree. Starting with a group of radical different trees, we modified each bracket slightly. When the probability of changing a bracket is greater then 0.5, our metric outperforms considerable a classical distance metric.

It would be interesting to analyze how this measures behave with respect to a *tree-edit distance*. Differently from the Parseval measure and ours, they are not based on the similarity between the yields of the nodes, but on the number of operations that are necessary to go from one tree to another one. The standard edit-distance [5, 14, 2] contemplates operations of insertion, deletion and renaming of nodes. In order to have a meaningful measure, an operation on edges that would permit to re-branch nodes (known as prune and regraft [3]) should also be considered.

Finally, we think that this measure is suitable to be used together with algorithms that output straight-line grammars to discover similarity that is not captured by comparing only the content of them.

Acknowledgements I am grateful to A. Apostolico and M. Comin for fruitful discussions.

References

1. Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., Strzalkowski, T.: Procedure for quantitatively comparing the syntactic coverage of english grammars. In: Black, E. (ed.) HLT '91: Proceedings of the workshop on Speech and Natural Language. pp. 306–311. Association for Computational Linguistics, Morristown, NJ, USA (1991)
2. Akutsu, T., Fukagawa, D., Takasu, A.: Approximating tree edit distance through string edit distance. Algorithmica 57(2), 325–348 (Jan 2010)
3. Allen, B., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Annals of combinatorics 5(1), 1–15 (2001)
4. Apostolico, A., Lonardi, S.: Off-line compression by greedy textual substitution. Proceedings of the IEEE 88, 1733–1744 (Jan 2000)
5. Bille, P.: A survey on tree edit distance and related problems. Theoretical Computer Science 337(1-3), 217–239 (Jan 2005)
6. Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, A.: The smallest grammar problem. IEEE Transactions on Information Theory 51(7), 2554–2576 (July 2005)
7. Jr, F.B.: Three great challenges for half-century-old computer science. Journal of the ACM 50(1) (Jan 2003)
8. Kieffer, J., Yang, E.H.: Grammar-based codes: a new class of universal lossless source codes. IEEE Transactions on Information Theory 46, 737–754 (2000)
9. Klein, D.: The Unsupervised Learning of Natural Language Structure. Ph.D. thesis, University of Stanford (2005)
10. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–87 (1955)
11. Lipkus, A.: A proof of the triangle inequality for the Tanimoto distance. J Math Chem 26(1-3), 263–265 (Jan 1999)
12. Nevill-Manning, C.G.: Inferring Sequential Structure. Ph.D. thesis, University of Waikato (1996)
13. Sampson, G.: A proposal for improving the measurement of parse accuracy. International J. of Corpus Linguistics 5, 53–68 (2000)

14. Touzet, H.: A linear tree edit distance algorithm for similar ordered trees. In: Combinatorial Pattern Matching. vol. 3537, pp. 334–345 (Jan 2005)