View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repositorio Institucional de la Universidad de Alicante

A distance for partially labeled trees

No Author Given

No Institute Given

Abstract. Trees are a powerful data structure for representing data for which hierarchical relations can be defined. It has been applied in a number of fields like image analysis, natural language processing, protein structure, or music retrieval, to name a few. Procedures for comparing trees are very relevant in many tasks where tree representations are involved. The computation of these measures is usually time consuming and different authors have proposed algorithms that are able to compute them in a reasonable time, by means of approximated versions of the similarity measure. Other methods require that the trees are fully labeled for the distance to be computed. The measure utilized in this paper is able to deal with trees labeled only at the leaves that runs in $O(|T_1| \times |T_2|)$ time. Experiments and comparative results are provided.

Key words: Tree edit distance, approximate distances, qtrees

1 Introduction

The computation of a measure of the similarity between two trees is a subject of interest in very different areas where trees are suitable structures for data coding. Trees are able to code hierarchical relations in their structure in a natural way and they have been utilized in many tasks, like text document analysis [10], protein structure [14], image representation and coding [3], or music coding and retrieval [12], to name just a few.

Different approaches have been posed in order to perform this comparison. Some of them impose a restriction on how the comparison is performed, while others establish valid mappings. Some methods pay more attention on the tree structure, and others pay it on the content of the nodes and the leaves. Most of them are designed to work with fully labeled trees.

The method proposed in this paper is designed to work with partially labeled trees, more precisely with those labeled only at the leaves. This fact, focus more on the coded content and the relations within its context. One of the fields where this situation is relevant is music comparison and retrieval. Trees have been used for this task and a number of representation and comparison schemes have been applied based on tree edit distances [12] or probabilistic similarity schemes [5].

In any case, the computation of these measures is usually a time consuming task and different authors have proposed algorithms that are able to compute them in a reasonable time [16], through approximated versions of the similarity measure. In this paper, a new algorithm is presented, able to deal with trees labeled only at the leaves that runs in $O(|T_1| \times |T_2|)$ time.

2 Tree comparison methods

In general, trees can be divided in *orderered* and *not ordered trees*, and in *evolutionary* and *not evolutionary* trees. In this work only ordered trees are considered. Regarding the *evolutionary trees*, they are often used to conceptually represent the evolutionary relationship of species or organisms in biology, evolution of works in linguistics, statistical classifications, or even tracking computer viruses. An *evolutionary tree* can be defined as a tree with distinct labels at leaves. Several algorithms have been given to solve the comparison of evolutionary trees [13, 2, 9]. In this work we deal with trees that have not distinct labels, and thus they are not *evolutionary trees*, so those algorithms are not applicable for our problem.

A number of similarity measures for ordered non-evolutionary trees have been defined in the literature [1]. Some of them measure the sequence of operations needed to transform one tree in another one, others look for the longest common path from the root to a tree node. There are methods that allow wildcards in the matching process in the so-called *variable-length doesn't care* (VLDC) distance. Several taxonomies have been proposed. The interested reader can look up a hierarchy of tree edit distances in [8] and [18].

Table 1: Some tree edit and alignment distance algorithms and their time complexities.

Tai [17]	$O(\lvert T\rvert \times \lvert T'\rvert \times \mathrm{depth}(T)^2 \times \mathrm{depth}(T')^2)$
Shasha & Zhang [19]	$O(\lvert T\rvert \times \lvert T'\rvert \times \min\{\mathrm{depth}(T), \lvert \mathrm{leaves}(T)\rvert\} \times \min\{\mathrm{depth}(T'), \lvert \mathrm{leaves}(T')\rvert\})$
Jiang [7]	$O(\lvert T\rvert \times \lvert T'\rvert \times (\mathrm{rank}(T) + \mathrm{rank}(T'))^2)$
Selkow [16]	$O(\lvert T\rvert \times \lvert T'\rvert)$
Valiente [18]	$O(\lvert T\rvert \times \lvert T'\rvert \times \log(\lvert T\rvert + \lvert T'\rvert))$

Some of the most relevant tree edit and alignment distances have been compiled in Table 1 together with their time complexities. These measures are the ones utilized for comparison in this paper, except that of Tai due to its very high complexity.

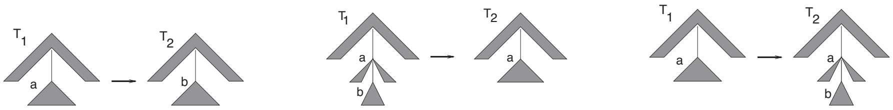
The classical edit distance between two trees $d(T, T')$ is computed using an edit script $e = e_1 \cdots e_n$ that is a sequence of edit operations allowing the transformation of a tree T to a tree T'. The cost of an edit script $C(e)$ is the sum of the costs of the edit operations involved in the script: $C(e) = \sum_{i=1}^{n} c(e_i)$. Given $S(T, T')$, the set of all the scripts that enable the emission of T' given T, the edit distance between T and T' is defined by: $d(T, T') = \min_{e \in S(T,T')} C(e)$. The edit operations allowed over two trees are any of the following:

– *relabel* the label l of a node v of T by the label l' of another node w of T', denoted by (v, w) (Fig. 1a).

– *deletion* of a non-root node v from T, denoted by (v, λ), consists of deleting it, making the children of v become the children of its parent node, in the position that was occupied by v, preserving this way the left to right ordering of leaves (Fig. 1b).

- *insertion* of a non-root node w in T', denoted by denoted by (λ, w). Given a sequence $w_i \cdots w_j$ of subtrees of a common parent w, the insertion of node w' makes those $w_i \cdots w_j$ subtrees children of w', and w' child of w (Fig. 1c).

Note that the operation (λ, λ) is not allowed.



(a) Substitute operation (b) Delete operation (c) Insert operation

Fig. 1: Tree edit operations (from [6]).

The edit cost of each operation, $c(e_i)$ is given based on that of the edit cost of the symbols for the labels, $c\colon \Sigma \times \Sigma \rightarrow \mathbb{R}$ that depends on the particular application. Therefore, $c(e_i)$ denotes the cost of applying the edit operation (v, w).

The approaches by Selkow [16], Valiente [18], and Jiang [7] are restricted version of this general methodology.

3 Proposed tree comparison algorithm

The edit distances presented above are designed to work with fully labeled trees. In order to apply those algorithms to trees labeled only at the leaves, the non-labeled inner nodes can be assigned a special label "empty". However, it is expected they don't work as well as they do with fully labeled trees.

In order to overcome this situation two approaches are possible. The first one consists of labeling all nodes using any bottom-up propagation scheme using specific knowledge of the application domain. The main drawback of that option is that any intermediate process will condition the resulting trees, with a loss of generality. The second approach is to design a distance function able to compare partially labeled trees.

The *partially labeled tree comparison algorithm* (s_p) is based on the assumption that the similarity value between a labeled leaf and a non-labeled inner node should be the average of the chances of finding that leaf in the descendants of that inner node. Fig. 2a shows the simplest case of having two leaf trees: $s_p(T_A, T_B) = \delta(a, b)$, where $\delta(a, b) = 1 \iff a = b$, and 0 elsewhere. For comparing the trees shown in Fig. 2b, the chances of finding the label a in T_B are computed as $s_p(T_A, T_B) = (\delta(a, x) + \delta(a, y))/2$. If, instead of a label, y another tree is placed there, the function should be computed recursively. Finally, when none of the trees is composed by a single leaf (Fig. 2c), the similarity function is computed like an edit distance between sequences wx and yz, where each symbol is a tree.

This measuring method omits the accounting of the insertion or deletion of nodes and only measures the chance of finding matching labels, giving more importance to the information hierarchically contained in the tree than to the tree structure.

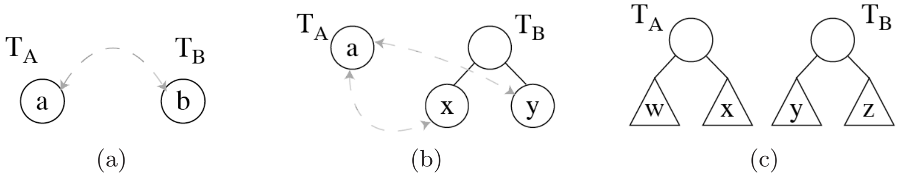


Fig. 2: Similarity function s_p representative cases.

This method is designed for working with partially labeled trees, but we can slightly adapt the original idea to work with fully labeled trees. The case of comparing a leaf to a non-leaf tree (Fig. 3a) is computed as $s_p(T_A, T_B) = (\delta(a, b) + \delta(a, x) + \delta(a, y))/3$. And in the same way as in the case of non-labeled nodes, the similarity $s_p(T_A, T_B)$ between two fully labelled trees (Fig. 3b) is computed as the edit distance between the sequences wx and yz, where each symbol is a tree, plus now the similarity between the labels a and b.

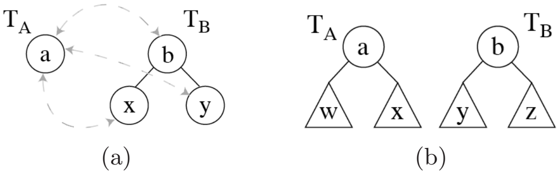


Fig. 3: Similarity function s_p working on fully labelled trees.

The algorithmic details of this method can be found at [11]. In that paper it is also shown that the time complexity of the proposed algorithm is $O(|T_1| \times |T_2|)$, like that fastest of the methods cited in Table 1.

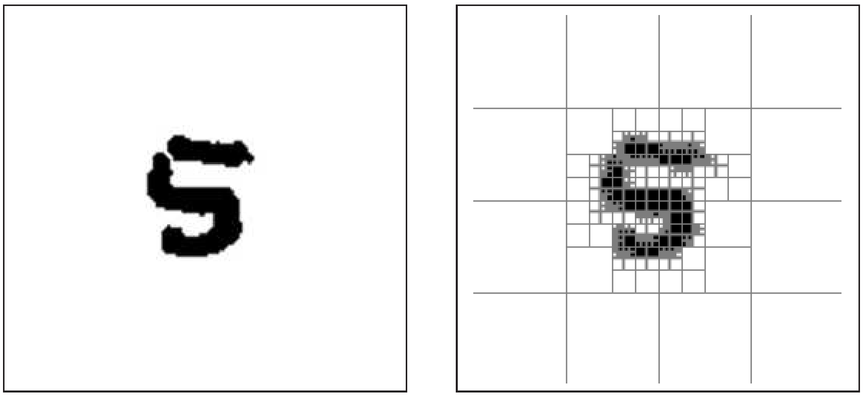
4 Experiments and results

The experiments are devised to show that the proposed tree distance is able to provide good results in a reasonable time when compared to other classical tree comparison algorithms. For that, we need to select an application where the data can be described in terms of trees.

In general, images can be coded as QuadTrees [15]. In a QuadTree, each inner node has four children. Each leaf represents a region in the image, labeled with any of its properties. If the considered property in the region covered by that leaf is not homogeneous enough (the deviation of their pixel values are over a threshold), the leaf is converted to a inner node and 4 leaves are created, splitting the region into 4 sub-regions. This procedure is repeated recursively until al the regions are homogeneous. At this point the tree is built, containing labels only at the leaves.

In particular, for binary images, the alphabet for the labels is $\Sigma = \{0, 1\}$. That is the case for the images of the isolated hand-written character images that have been used and coded as QuadTrees (Fig. 4) for our experiments. The problem to solve is to identify a character or digit from a set made by different writers.

It should be pointed out that this evaluation is not oriented to outperform the state of the art in hand-written OCR, but to check whether the proposed tree edit algorithm is useful for solving tree-encoded structure comparisons efficiently. In order to this, its performance has been compared to those of other tree distances, not to other handwritten OCR methods.



(a) Original image (b) Image encoded as QuadTree

Fig. 4: Encoding image as a QuadTree.

Corpora. The NIST image gallery [4] was utilized. To code the character images with QuadTrees the usual procedure has been followed, splitting recursively the image in four parts until they reach a level where all the pixels in each new part have the same value (foreground or background). That value will be placed only in the leaves.

Two corpora have been used in our experimental setup. The first consists of a set of 19,540 digits, the same number for each one. The second corpus consists of a set of 50,400 letter-and-digits images, with the same number for each class again.

Character classification accuracy The experiments have been performed using the following scheme. The corpora have been split into 10 folds. Then, given each fold, for each class, a prototype has been taken as a query, being compared to all prototypes in the fold, obtaining a list L of prototypes sorted by similarity value, L_k being the k-th position in the list.

The accuracy of the system has been computed using the *mean reciprocal rank* (MRR). Let q be a query, p a prototype, and let the function $c(p)$ return the class of the prototype p, the reciprocal rank RR for a query is computed as:

$$RR(q) = \frac{1}{\arg\min_k \{c(L_k(q)) = c(q)\}} \tag{1}$$

The denominator should be read as the minimum position in the list, k, so that the predicted class matches that of the query.

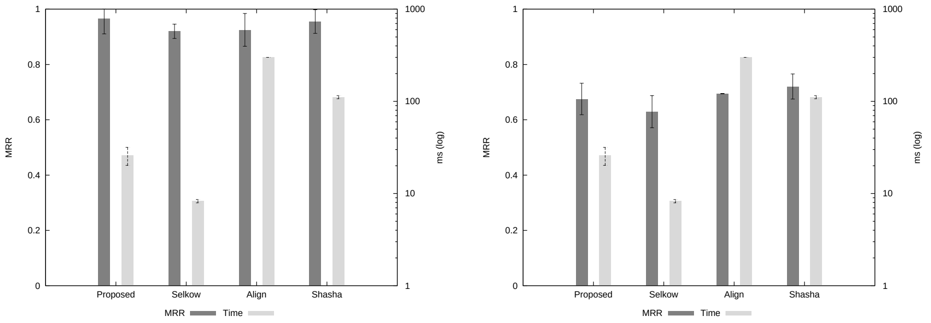
The *mean reciprocal rank* is the mean of all $RR(q)$ for all the queries.

Running times were measured in milliseconds, taking into account only the test phase, leaving aside the construction of the representations that may be done off-line. All experiments were performed using a Sun machine with 8 Gb RAM and 8 Intel(R) Xeon(R) CPU X5355 running at 2.66GHz, with a SUSE Linux with kernel version 2.6.

4.1 Results

The achieved results have been shown in the Fig. 5. For each tree comparison method the MRR has been extracted to be used as y-axis in the bars. The processing time for each query has also been used as the other y-axis.

The results plotted in Fig. 5a denote that the proposed algorithm achieves a success rate comparable to the best algorithms but the processing time is almost as efficient as the faster method. When working with alphanumeric corpus (Fig. 5b), the proposed algorithm behaves in a similar level of performance regarding the other methods.



(a) Digits corpus

(b) Alphanumeric corpus

Fig. 5: Success rates of proposed method compared to classical tree edit distances.

Thus, it seems that the proposed algorithm is able to compare hand-written character images encoded as QuadTree better than the other classical tree comparison algorithms in terms of trade-off between time and success rate.

5 Conclusions

An approximate tree edit distance algorithm has been introduced for working with trees labeled only at the leaves. In order to assess its performance, it has been applied to handwritten character recognition using the well-known NIST database. Our aim was not to outperform the state of the art but to show that the proposed tree distance is able to provide good results in a reasonable time when compared to other tree distances.

The performance has been compared to classical tree similarity algorithms from the literature. The results show that, in the used context, the proposed algorithm achieved accuracies comparable to those by the most comprehensive search method and it is as efficient as the fastest.

References

1. Philip Bille. A survey on tree edit distance and related problems. *Theorical Computer Science*, 337(1-3):217–239, 2005.
2. B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On distances between phylogenetic trees. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 427–436, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.
3. Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.
4. M.D. Garris and R.A. Wilkinson. Nist special database 3: Handwritten segmented characters. *NIST, Gaithersburg, Md.*
5. Amaury Habrard, José M.. Iñesta, David Rizo, and Marc Sebban. Melody recognition with learned edit distances. *Lecture Notes in Computer Science*, 5342:86–96, 2008.
6. Carsten Isert. The editing distance between trees, 1999.
7. Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees – an alternative to tree edit. *Theoretical Computer Science*, 143(1):137 – 148, 1995.
8. Tetsuji Kuboyama, Kilho Shin, and Tetsuhiro Miyahara. A hierarchy of tree edit distance measures. *Theoretical Computer Science and its Applications*, 2005.
9. Chuan-Min Lee, Ling-Ju Hung, Maw-Shang Chang, Chia-Ben Shen, and Chuan-Yi Tang. An improved algorithm for the maximum agreement subtree problem. *Information Processing Letters*, 94(5):211 – 216, 2005.
10. Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert Macintyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *In ARPA Human Language Technology Workshop*, pages 114–119, 1994.
11. Authors ommited for blind review. New partially labelled tree similarity measure: a case study. *Lecture Notes in Computer Science*, 6218:296–305, august 2010.
12. David Rizo, Kjell Lemström, and José 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*, 5493/2009:177–195, 2009.
13. D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1-2):131–147, February 1981.
14. Robert B. Russell and Geoffrey J. Barton. Multiple protein sequence alignment from tertiary structure comparison: Assignment of global and residue confidence levels. *Proteins: Structure, Function, and Bioinformatics*, 14:309–323, 2004.
15. Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

16. Stanley M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.
17. Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.
18. G. Valiente. An efficient bottom-up distance between trees. In *International Symposium on String Processing and Information Retrieval*, pages 212–219, Los Alamitos, CA, USA, 2001. IEEE Computer Society.
19. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.