épisciences

Accurate comparison of tree sets using HMM-based descriptor vectors

Sylvain Iloga[*1,2,3]

[1]Higher Teachers' Training College, University of Maroua, P.O.box 55 Maroua, Cameroon
[2]IRD, UMMISCO F-93143, Sorbonne University, F-93143 Bondy, France
[3]ENSEA, CNRS, ETIS UMR 8051, CY Cergy Paris University, F-95000 Cergy, France

*E-mail : sylvain.iloga@gmail.com

DOI : 10.46298/arima.9107
Submitted on February 21, 2022 - Published on July 26, 2022
Volume : **36** - Year : **2022**
Special Issue : **Special issue CRI 2021**
Editors : *René Ndoundam, Eric Badouel, Maurice Tchuenté, Paulin Melatagia*

Abstract

Trees are among the most studied data structures and several techniques have consequently been developed for comparing two trees belonging to the same category. Until the end of year 2020, there was a serious lack of suitable metrics for comparing two weighted trees or two trees from different categories. The problem of comparing two tree sets was not also specifically addressed. These limitations have been overcome in a paper published in 2021 where a customizable metric based on hidden Markov models has been proposed for comparing two tree sets, each containing a mixture of trees belonging to various categories. Unfortunately, that metric does not allow the use of non metric-dependent classifiers which take descriptor vectors as inputs. This paper addresses this drawback by deriving a descriptor vector for each tree set using meta-information related to its corresponding models. The comparison between two tree sets is then realized by comparing their associated descriptor vectors. Classification experiments carried out on the databases *FirstLast-L* (FL), *FirstLast-LW* (FLW) and *Stanford Sentiment Treebank* (SSTB) respectively showed best accuracies of 99.75%, 99.75% and 87.22%. These performances are respectively 40.75% and 20.52% better than the tree Edit distance for FLW and SSTB. Additional clustering experiments exhibited 54.25%, 98.75% and 75.53% of correctly clustered instances respectively for FL, FLW and SSTB. No clustering was performed in existing work.

Keywords

Trees; Comparison of tree sets; Distance between trees; Hidden Markov Models; tree Edit distance; Classification; Clustering

I INTRODUCTION

Trees are among the most common and well-studied data structures. They are used in a great variety of applications, including compiler design, graph transformation, automatic theorem

proving, information retrieval, structured text database, pattern recognition, and signal processing [1, 2]. Consequently, several techniques have been developed for comparing two labeled or unlabeled trees belonging to the same category (i.e: rooted or unrooted, ordered or unordered). Until the end of year 2020, these existing techniques were based on one of the 5 following concepts: *tree Edit* [3–14], *tree Alignment* [15, 16], *tree Inclusion* [17–21], *tree pattern matching* [22–27] and *Subtrees/supertrees similarity* [28–34]. These existing techniques unfortunately have the three following main drawbacks:

1. There is a serious lack of suitable metrics for comparing two weighted trees, and yet the comparison of weighted trees is important for several applications such as the matchmaking of agents in e-Business environments where product/service descriptions and seller/buyer agents are represented as arc-weighted trees [35]. An attempt for comparing such trees was proposed in [35], but that measure was limited because arc weights were restricted to the interval $[0, 1]$.
2. Existing techniques did not enable to compare two trees from different categories (e.g: comparing a rooted ordered tree and an unrooted unlabeled tree).
3. The problem of comparing two tree sets was not specifically addressed. However, the comparison of tree sets is important for several machine learning tasks including the *hierarchical clustering of tree data*. Indeed, at each step of hierarchical clustering algorithms like the *Agglomerative Hierarchical Clustering* (*AHC*) algorithm [36], the distance between every pair of clusters (tree sets) is computed in order to merge the two nearest clusters into a new cluster. Existing cluster distances like the *minimum*, the *maximum*, the *average* or the *centroid* linkage distances [37] are generally used for this purpose. But such an approach does not consider the individual properties of the trees of each cluster.
4. Existing techniques did not enable to explicitly specify the targeted node properties on which the tree comparison must be performed.

In 2021, these drawbacks have been overcome in [38] where a customizable metric based on Hidden Markov Models (HMMs) was proposed for comparing two tree sets, each containing a mixture of trees belonging to various categories. The metric proposed in that paper handles labeled and unlabeled trees, as well as weighted and unweighted trees where each node/arc can have several attributes (labels, weights). Unlike previous techniques, it allows the user to explicitly specify the targeted node properties on which the comparison should be performed and there is no restriction on these properties. In that paper, one relies on the *Depth-First Search* (*DFS*) traversal of a tree [39] to design the HMMs. The comparison between two tree sets is finally performed by comparing their associated HMMs. That metric shows a perfect accuracy of 100% during flat classification experiments carried out on the two synthetic databases [1] *FirstLast-L* (FL) and *FirstLast-LW* (FLW) using the *Nearest Neighbor* (*NN*) classifier. This performance is 41% higher than the one exhibited when the *tree Edit* distance is selected for FL.

An important limitation of the metric proposed in [38] is related to the fact that it does not allow the use of non metric-dependent classifiers (i.e: *Support Vector Machines* [40], *Decision trees* [41], etc.) which require descriptor vectors as inputs. The current paper addresses this limitation by deriving a descriptor vector for each tree set from its associated models. Consider the set $P = \{p_1, \ldots, p_m\}$ of targeted node properties. Given a node property $p_i \in P$ and the model $\lambda_i(T)$ associated with a tree set T according to p_i, we capture the overall proportion of time spent by $\lambda_i(T)$ observing each symbol after a sufficiently long time, irrespective of the state from which this symbol is observed as a characteristic of T. We apply this for all the properties

[1] http://perso-etis.ensea.fr/sylvain.iloga/FirstLast/index.html

in P and the resulting values are sequentially gathered as the components of a descriptor vector $\overrightarrow{T}$. Finally, the comparison between two tree sets is realized by comparing their associated descriptor vectors. The relevance of the proposed descriptor vectors is illustrated through classification and unsupervised clustering experiments carried out on two synthetic tree databases and one real world tree database.

The rest of this paper is organized as follows: the state of the art on tree comparison is presented in Section II, followed by a summarized presentation of HMMs in Section III. The description of the proposed approach is realized in Section IV, while experimental results are exhibited in Section V. The last section is dedicated to the conclusion.

II RELATED WORK AND PROBLEM STATEMENT

2.1 Related work

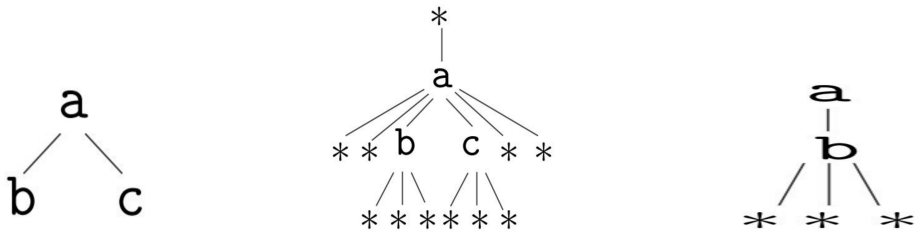
A tree can be defined as a connected acyclic graph. A detailed overview on graph theory and applications is available in [42]. Many distances have yet been proposed for comparing trees and they are based on one of the following concepts: *tree edit*, *tree alignment*, *tree inclusion*, *tree pattern matching*, *subtrees/supertrees similarity* and *HMMs*.

The *tree edit* distance [3–14] is based on the analysis of the number of edit operations required to transform a tree t_1 into another tree t_2. The three following edit operations are considered for a given node: insertion, deletion and substitution. *Tree alignment* [15, 16] is a particular case of *tree edit* where insertions are always performed before deletions. Let t_1 and t_2 be two rooted trees with labeled nodes. t_1 is *included* in t_2 if there is a sequence $S(t_1, t_2)$ of node deletions performed on t_2 which makes t_2 isomorphic to t_1. The tree inclusion problem is to decide if t_1 can be included in t_2 and the *tree inclusion* distance [17–21] is the sum of the costs of the delete operations found in $S(t_1, t_2)$. *Tree pattern matching* [22–27] consists in finding the instances of a given pattern tree in a specific target tree. *Subtrees and supertrees similarity* [28–33] are generally realized by finding the maximum agreement subtree, the largest common subtree or the smallest common supertree.

Given a tree t and two positive user-defined integers p, q, the *pq-extended tree* $t^{p,q}$ is constructed from t by [2]: (1) adding $(p-1)$ ancestors to the root node, (2) inserting $(q-1)$ children before the first and after the last child of each non-leaf node, (3) adding q children to each leaf node. All newly inserted nodes are dummy nodes that do not occur in t and have the special label $*$. Figure 1(b) shows the $(2,3)$-extended tree $t^{2,3}$ derived from the node-labeled tree t depicted in Figure 1(a). A subtree of $t^{p,q}$ is a pq-gram g of t if and only if [3]: (1) g has q leaf nodes and p non-leaf nodes, (2) all leaf nodes of g are the children of a single node, (3) the leaf nodes of g are consecutive siblings in $t^{p,q}$. Figure 1(c) shows a $(2,3)$-gram g of the node-labeled tree t depicted in Figure 1(a). One can compare two trees by computing their *pq-gram distance* which is the number of pq-grams that are not shared by the two trees [34] [4].

Until the end of year 2020, existing techniques for comparing two trees [3–34] embedded many drawbacks which were overcome in 2021 by the customizable HMM-based metric proposed in [38] for comparing two tree sets. This *HMM*-based technique is based on the fact that, as *DFS* sequentially transits from one node depth to another and at each step, one can observe the

[2] See Definition 1 of [34]
[3] See Definition 2 of [34]
[4] See Section 3.3 of [34]



((a)) The tree t ((b)) The tree $t^{2,3}$ ((c)) The tree g

Figure 1: Derivation of a $(2, 3)$-gram g of a tree t

properties of the visited node. Consider the set $P = \{p_1, \ldots, p_m\}$ of targeted node properties. For every property $p_i \in P$, the observation above enables to transform any tree t into a Markov chain $\delta_i(t)$ where the hidden states are the node depths, while the symbols are the values of property p_i for the nodes of t. When this principle is applied to a tree set $T = \{t_1, \ldots, t_n\}$, the set $\Delta_i(T) = \{\delta_i(t_1), \ldots, \delta_i(t_n)\}$ of Markov chains is obtained. The content of $\Delta_i(T)$ is later used to initialize and train the HMM $\lambda_i(T)$ associated with T according to property p_i. Finally, the comparison between two tree sets is performed through the comparison of their associated HMMs for every property $p_i \in P$. The main assets of that technique are listed below:

1. It is designed for comparing finite tree sets.
2. It handles rooted/unrooted as well as ordered/unordered trees.
3. It compares weighted/unweighted as well as labeled/unlabeled trees.
4. It allows each node/arc to have many labels/weights.
5. It requires the specification of the targeted nodes properties.
6. It outperforms the *tree edit* distance with 41% of accuracy gain.

All the distances between two trees reviewed in this section can be used for performing the hierarchical clustering of tree data which involves the comparison of tree sets (clusters) as stated at the paper Introduction.

2.2 Problem statement

An important limitation of that technique is related to the fact that it does not allow the use of non metric-dependent classifiers which absolutely require descriptor vectors as inputs. The current paper tackles this limitation by deriving a descriptor vector for each tree set from its associated models. The comparison between two tree sets is then realized by comparing their respective associated descriptor vectors.

III HIDDEN MARKOV MODELS

3.1 HMM definition

A HMM $\lambda = (A, B, \pi)$ is fully characterized by [43]:

1. The number N of states of the model. The set of states is $S = \{s_1, s_2, \ldots, s_N\}$. The state of the model at time x is generally noted $q_x \in S$.
2. The number M of symbols. The set of symbols is $\vartheta = \{v_1, v_2, \ldots, v_M\}$. The symbol observed at time x is generally noted $o_x \in \vartheta$.
3. The state transition probability distribution $A = \{A[s_i, s_j]\}$ where $A[s_i, s_j] = Prob(q_{x+1} = s_j | q_x = s_i)$ with $1 \leq i, j \leq N$.

4. The symbols probabilities distributions $B = \{B[s_i, v_k]\}$ where $B[s_i, v_k] = Prob(v_k \text{ at time } x | q_x = s_i)$ with $1 \leq i \leq N$ and $1 \leq k \leq M$.
5. The initial state probability distribution $\pi = \{\pi[s_i]\}$ where $\pi[s_i] = Prob(q_1 = s_i)$ with $1 \leq i \leq N$.

3.2 Manipulation of a HMMs

Consider a sequence of symbols $O = o_1 o_2 \ldots o_X$ and a HMM $\lambda = (A, B, \pi)$. The probability $Prob(O|\lambda)$ to observe O given λ is efficiently calculated by the *Forward-Backward* algorithm [43] which runs in $\theta(X.N^2)$. Given a sequence of symbols $O = o_1 o_2 \ldots o_X$, it is possible to iteratively re-estimate the parameters of a HMM $\lambda = (A, B, \pi)$ in order to maximize the value of $Prob(O|\overline{\lambda})$, where $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ is the re-estimated model. The *Baum-Welch* algorithm [43] is generally used to perform this re-estimation. This algorithm runs in $\theta(\beta.X.N^2)$ where β is the user-defined maximum number of iterations. The *Baum-Welch* algorithm can also train a HMM for multiple sequences. The algorithm maximizes the value of $Prob(O|\overline{\lambda}) = \sum_{k=1}^{K} Prob(O^{(k)}|\overline{\lambda})$ where $O = \{O^{(1)}, \ldots, O^{(K)}\}$ is a set of K sequences of symbols and $O^{(k)} = o_1^{(k)} \ldots o_{X_k}^{(k)}$ is the k^{th} sequence of symbols of O. In the case of multiple sequences, this algorithm runs in $\theta\left(\beta.(\sum_{k=1}^{K} X_k).N^2\right)$. In this paper, the value $\beta = 100$ is selected following [38].

3.3 Stationary distribution of a HMM

A vector $\varphi = (\varphi[s_1], \ldots, \varphi[s_N])$ is a stationary distribution of a HMM $\lambda = (A, B, \pi)$ if [44] [5]:

1. $\forall j, (\varphi[s_j] \geq 0)$ and $\left(\sum_j \varphi[s_j] = 1\right)$
2. $\varphi = \varphi.A \Leftrightarrow (\varphi[s_j] = \sum_i \varphi[s_i] \times A[s_i, s_j], \forall j)$

$\varphi[s_j]$ estimates the overall proportion of time spent by λ in state s_j after a sufficiently long time. φ can be extracted from any line of the matrix $A^w = A \times A \times \ldots \times A$ (w times) when $w \to +\infty$. Therefore, the computation of φ requires $\theta(w.N^3)$ arithmetic operations.

IV THE PROPOSED APPROACH

4.1 Main idea

Consider the set $P = \{p_1, \ldots, p_m\}$ of targeted nodes properties and let us assume that the *DFS* traversal of a tree t is executed by a robot r. For each property $p_i \in P$, the robot r sequentially transits from one node depth to another and at each step, r can observe the value $p_i(y)$ of the currently visited node $y \in t$ [38]. The MC $\delta_i(t)$ resulting from this traversal of t executed by the robot r according to property p_i embeds relevant information related to the overall behavior (movements, observations) of r. Given a tree set $T = \{t_1, \ldots, t_n\}$ and its corresponding set $\Delta_i(T) = \{\delta_i(t_1), \ldots, \delta_i(t_n)\}$ of MCs, the HMM $\lambda_i(T)$ is trained with the *Baum-Welch* algorithm to learn and to simulate the overall behavior of r during the traversal of the trees in T.

The main idea of the current paper is that, one can capture the behavior of r from $\lambda_i(T)$ by evaluating the overall proportion of time spent by r observing each possible value of p_i after a sufficiently long time, irrespective of the node depth from which this value is observed. More precisely, we evaluate the probability $\gamma(o|\lambda_i(T))$ of observing a symbol o after a sufficiently

[5] See Definition 9.1 of [44]

long time, given the model $\lambda_i(T)$, irrespective of the state from which o is observed. When this principle is applied for every property $p_i \in P$ and for every possible symbol, the resulting probabilities are sequentially saved as the components of the descriptor vector $\overrightarrow{T}$ associated with the tree set T.

4.2 Method

Consider a tree set $T = \{t_1, \ldots, t_n\}$ and the set $P = \{p_1, \ldots, p_m\}$ composed of m user-defined targeted node properties. The method applied in the current paper for deriving the descriptor vector $\overrightarrow{T}$ associated with T can be summarized in the three following steps:

1. **Tree modeling:** For each property $p_i \in P$, the tree modeling principle proposed in [38] [6] is used to obtain the model $\lambda_i(T)$. Therefore, this step is not described in the following sections.
2. **Probability computing:** Let ϑ_i be the set of symbols of $\lambda_i(T)$. For every property p_i and for every symbol $o \in \vartheta_i$, we compute the probability $\gamma(o|\lambda_i(T))$ of observing o after a sufficiently long time, given the model $\lambda_i(T)$, irrespective of the state from which o is observed.
3. **Construction of the descriptor vector:** All the probabilities computed at step 2 are sequentially saved as the components of the descriptor vector $\overrightarrow{T}$ associated with T. Figure 2 summarizes the main steps of the proposed method.

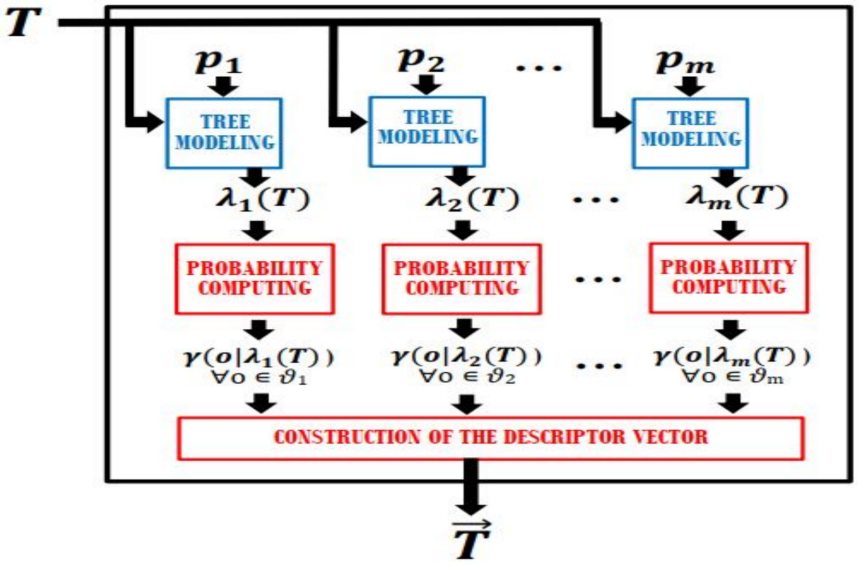


Figure 2: Method for deriving the descriptor vector $\overrightarrow{T}$ associated with a tree set T.

4.3 Probability computing

Consider a tree set T and its associated model $\lambda_i(T) = (A_i^T, B_i^T, \pi_i^T)$ according to property $p_i \in P$, where $P = \{p_1, \ldots, p_m\}$ is the user-defined set of targeted nodes properties. In order to compute $\gamma(o|\lambda_i(T))$, one must first evaluate the overall proportion of time spent by $\lambda_i(T)$ observing o in state s_j after a sufficiently long time as follows:

1. Evaluate the overall proportion of time spent by $\lambda_i(T)$ in state s_j after a sufficiently long time. This proportion is given by the j^{th} component $\varphi_i^T[s_j]$ of the stationary distribution of $\lambda_i(T)$.

[6] See Section 4.3 and Figure 4 of [38]

2. Multiply the result of step 1 by the probability of observing o in state s_j which is $B_i^T[s_j, o]$. The value of $\gamma(o|\lambda_i(T))$ is finally obtained by repeating this process for every state s_j and summing the resulting proportions as shown in Equation 1. An analog probability computing scheme was performed in [45] where the authors used HMMs for realizing human activity recognition [7].

$$\gamma(o|\lambda_i(T)) = \sum_{j=1}^{N} \varphi_i^T[s_j] \times B_i^T[s_j, o] \tag{1}$$

4.4 Construction of the descriptor vector

The descriptor vector $\overrightarrow{T}$ associated with T is constructed by sequentially saving the values of $\gamma(o|\lambda_i(T))$ for every property $p_i \in P$ and for every symbol $o \in \vartheta_i$ as described in Algorithm 1. The first line of Algorithm 1 initializes the index of the components of the descriptor vector. Line 2 browses the properties, while line 3 browses the symbols for each property. The value of $\gamma(o|\lambda_i(T))$ is computed at line 4 according to Equation 1 and saved as the current component of the descriptor vector at line 5. Line 6 moves to the index of the next component of the descriptor vector. The descriptor vector is finally returned at line 9.

Algorithm 1 *Vector*

Inputs: $T, P = \{p_i\}, \{\lambda_i(T)\}, \{\vartheta_i\}, \{\varphi_i^T\}$ with $(1 \leq i \leq m)$
Output: $\overrightarrow{T}$

1: $k \leftarrow 1$
2: **for each** $(p_i \in P)$ **do**
3: **for each** $(o \in \vartheta_i)$ **do**
4: $\gamma(o|\lambda_i(T)) \leftarrow \sum_{j=1}^{N} \varphi_i^T[s_j] \times B_i^T[s_j, o]$
5: $\overrightarrow{T}[k] \leftarrow \gamma(o|\lambda_i(T))$
6: $k \leftarrow k + 1$
7: **end for**
8: **end for**
9: **return** $\overrightarrow{T}$

A consequence of Algorithm 1 is that the dimension of the descriptor vector $\overrightarrow{T}$ denoted in this paper as α is the sum of the numbers of symbols of all its $|P|$ associated models as shown in Equation 2. It can be easily demonstrated that Algorithm 1 requires $\theta(\alpha.N)$ arithmetic operations.

$$\alpha = \sum_{i=1}^{|P|} |\vartheta_i| \tag{2}$$

The following conventional property is adopted here to associate a descriptor vector with the empty set: '*The descriptor vector associated with* $\emptyset$ *according to the set* $P = \{p_1, \ldots, p_m\}$ *of targeted nodes properties is* $(\overrightarrow{\emptyset} = \overrightarrow{0} \in \mathbb{R}^{\alpha})$.'

[7] See Section IV-E of [45]

4.5 Comparison of tree sets

The comparison between two tree sets T_1 and T_2 according to the set $P = \{p_1, \ldots, p_m\}$ of targeted nodes properties is performed here by calculating the value of any existing distance or similarity measure between their associated descriptor vectors $\overrightarrow{T_1}$ and $\overrightarrow{T_2}$. Equation 3 shows how this can be done with the *Euclidean* distance and the *Manhattan* distance. Several other possible distance and similarity measures are available online [8].

$$\begin{aligned} d_P(T_1, T_2) &= \sqrt{\sum_{k=1}^{\alpha} (\overrightarrow{T_1}[k] - \overrightarrow{T_2}[k])^2} \qquad \textit{(Euclidean)} \\ d_P(T_1, T_2) &= \sum_{k=1}^{\alpha} |\overrightarrow{T_1}[k] - \overrightarrow{T_2}[k]| \qquad \textit{(Manhattan)} \end{aligned} \tag{3}$$

V EXPERIMENTAL RESULTS

5.1 Classification and clustering algorithms

The flat classification and clustering experiments realized in the current work are both realized with the *WEKA* software [46]. Unlike [38] where only the *Nearest Neighbor* classifier was experimented, two additional non metric-dependent classifiers have also been experimented here. Thus, the three following classifiers have been experimented, their corresponding names in WEKA are in brackets: *Nearest Neighbor* (IBk), *Support Vector Machines* with polynomial kernels (SMO) and *Decision Trees* (J48). These classifiers have been used in WEKA with their default settings and a 10 fold cross-validation ($90\% - 10\%$) has been applied for each experiment.

The K*means* [47] and the *Expectation-Maximization* (EM) [48] clustering algorithms have been selected here to evaluate to what extent the proposed descriptor vectors can enable to accurately organize the experimental data (trees) into their corresponding classes (clusters) in an unsupervised process. The selected clustering algorithms have been used in *WEKA* with their default settings except the desired number of clusters which was initially fixed to the number of classes in each database.

The *Euclidean* and the *Manhattan* distances have been both experimented as metrics for the *Nearest Neighbor* classifier and the K*means* clustering algorithm.

[8] https://numerics.mathdotnet.com/Distance.html

5.2 Databases

The two synthetic tree databases selected during the experiments of the current work are FL and FLW [1] which have both been constructed and experimented in [38]. Each database is composed of 4 classes, each containing 100 rooted ordered trees. FL contains node-labeled trees and the set of node labels is $\{1, 2, \ldots, 26\}$. FLW contains the same trees found in FL, with weighted arcs. The trees in these two databases are characterized by non-trivial properties [9].

The *Stanford Sentiment Treebank* (SSTB) tree database has also been used in the current work. This is a real world tree database originally proposed in [49] and containing 11.855 node-labeled binary trees associated with sentences from movie reviews organized into 5 classes according to sentiments: *very negative* (1510 trees), *negative* (3140 trees), *neutral* (2242 trees), *positive* (3111 trees) and *very positive* (1852 trees). These classes are considered as node labels represented as integers belonging to $\{0, 1, 2, 3, 4\}$. In this database, each tree has the following recursive brackets representation: *(label (left subtree)(right subtree))* where the leftmost label is the overall sentiment of the sentence. Leaf nodes are of the form *(label token)* where token is a word of the sentence. During the experiments, the label of the root node of each tree was replaced by a new unique label (here 5) and tokens were removed from all the leave nodes following [34].

5.3 Experimental settings

We realized 2 main experiments for the databases FL and FLW. During the first one, the specific properties verified by each node in FL and FLW are intentionally ignored. During the second main experience, these specific properties are considered. The sets $\overline{P}_L = \{\overline{p}_1, \overline{p}_2\}$, $\overline{P}_{LW} = \{\overline{p}_1, \overline{p}_2, \overline{p}_3\}$, $P_L = \{p_1, \ldots, p_8\}$ and $P_{LW} = \{p_1, \ldots, p_{12}\}$ of targeted node properties selected in [38] [10] for these experiences are preserved in the current work.

The 3 following sub-datasets of SSTB have been used and Table 1 presents the size of each class in these sub-datasets:

1. **SSTB5**: all the 5 original classes of sentiments are considered.
2. **SSTB3**: the content of the original classes *very negative* and *negative* (resp. *very positive* and *positive*) are merged to obtain one single class named *negative* (resp. *positive*). Thus, three classes are considered.
3. **SSTB2**: This is SSTB3 where the class *neutral* is removed. Therefore, only two classes are considered here.

Table 1: Size of each class in the experimental sub-datasets of SSTB

	Very negative	**Negative**	**Neutral**	**Positive**	**Very positive**
SSTB5	1510	3140	2242	3111	1852
SSTB3	4650		2242	4963	
SSTB2	4650		0	4963	

Only the default set $\tilde{P} = \{\tilde{p}_1, \tilde{p}_2\}$ of targeted nodes properties has been experimented for the database SSTB, irrespective of its considered sub-dataset. For every node y of a tree in SSTB, $\tilde{p}_1(y)$ is the degree of y and $\tilde{p}_2(y)$ is the label of y. A detailed description of the experiments realized on the database SSTB and the corresponding data are available online [11].

[9] See Table 5 of [38]
[10] See Section 5.2.2 of [38]
[11] http://perso-etis.ensea.fr/sylvain.iloga/index.html

In the context of each experiment and considering each tree t of the experimental databases as the singleton $\{t\}$, we have executed Algorithm 1 to construct the descriptor vector $\overrightarrow{t}$ associated with t. The resulting descriptor vectors have then been saved into online available '*arff*' files [1,11] which are taken as inputs by the soft *WEKA*.

5.4 Classification results

Classification results are presented in Tables 2a, 2b and 2c respectively for FL, FLW and SSTB. Tables 2a and 2b reveal that best accuracy obtained for FL and FLW is 99.75%. This is quite close to the 100% obtained in [38]. Table 2c reveals that the proposed approach also exhibits good performance for the database SSTB with accuracies always greater than 70% for the two non metric-dependent classifiers applied to SSTB3 and SSTB2, the best accuracy being 87.22%. This performance is 20.52% better than the 66.7% obtained in [14] for SSTB2 [12]. These performances demonstrate the relevance of the proposed approach for classification, even with a default set of targeted node properties.

Table 2: Classification results. Accuracies are in $(\%)$, best values are in Bold.

(a)- *FL*

	IBk		*SMO*	*J48*
	Eucli.	*Manha.*		
$\overline{P}_L$	73.75	78	67.5	74.75
P_L	**99.75**	**99.75**	87.5	93.75

(b)- *FLW*

	IBk		*SMO*	*J48*
	Eucli.	*Manha.*		
$\overline{P}_{LW}$	84	87.25	90.75	89.75
P_{LW}	**99.75**	**99.75**	**99.75**	99.5

(c)- *SSTB* using $\tilde{P}$

	IBk		*SMO*	*J48*
	Eucli.	*Manha.*		
SSTB5	39.47	39.67	46.29	44.98
SSTB3	63.11	63.27	70.65	77.22
SSTB2	82.98	83.16	**87.22**	87.17

5.5 Clustering results

Clustering results are presented in Tables 3a, 3b and 3c respectively for FL, FLW and SSTB. Tables 3a and 3b reveal that best clustering performance obtained for FL is 54.25%, while a quasi perfect performance of 98.75% is obtained for FLW. Given that the trees in FL are only characterized by properties related to the topology and the node-labels, these results show that this database does not embed enough information to distinguish the 4 classes during an unsupervised clustering process. But when the arc-weights are considered in FLW, the 4 classes are accurately identified during the unsupervised clustering process with up to 98.75% correctly clustered instances when the 12 properties in P_{LW} are considered.

Table 3c reveals that the K*means* clustering algorithm applied with the *Manhattan* distance always exhibits the best clustering results for SSTB5, SSTB3 and SSTB2 which have respectively 33.33%, 49.02% and 75.53% correctly clustered instances. These performances illustrate the accuracy of the proposed approach in clustering, even with a default set of targeted nodes properties.

[12] See Table 1 of [14]

Table 3: Clustering results. Correctly clustered instances are in (%), best values are in Bold.

(a)- *FL*

	Kmeans		*EM*
	Eucli.	*Manha.*	
$\overline{P}_L$	47.5	47.25	**54.25**
P_L	45.5	44	42

(b)- *FLW*

	Kmeans		*EM*
	Eucli.	*Manha.*	
$\overline{P}_{LW}$	54.25	55.25	42.25
P_{LW}	86.5	92	**98.75**

(c)- *SSTB* using $\tilde{P}$

	Kmeans		*EM*
	Eucli.	*Manha.*	
SSTB5	32.63	33.33	29.17
SSTB3	40.43	49.02	46.36
SSTB2	50.97	**75.53**	61.73

5.6 Time cost

The time cost of the proposed technique for deriving the descriptor vector $\overrightarrow{T}$ of a tree set T according to the set P of targeted node properties can be estimated as follows:

1. Design the $|P|$ HMMs associated with T which runs in $\theta\left(|P|.\beta.(\sum_{t\in T}|t|).N^2\right)$ according to [38] [13], where $|t|$ is the number of nodes in the tree t.
2. Compute the stationary distributions of the $|P|$ HMMs which runs in $\theta(|P|.w.N^3)$ with $w\rightarrow+\infty$. In the current work, the value $w = 100$ has been selected.
3. Construct the descriptor vector $\overrightarrow{T}$ associated with T using Algorithm 1 which runs in $\theta(\alpha.N)$.

Equation 4 gives the expression of the theoretical time cost $Time(\overrightarrow{T})$ of the proposed approach.

$$\begin{aligned} Time(\overrightarrow{T}) &= \theta(a.N^3 + b.N^2 + \alpha.N) \text{ where} \\ N &= \max\{depth(t) \mid t\in T\} + 1 \\ a &= |P|.w \\ b &= |P|.\beta.\left(\sum_{t\in T}|t|\right) \end{aligned} \tag{4}$$

5.7 Comparisons with existing techniques

We have compared the proposed approach with:

1. [38] where the authors designed in 2021 the databases FL and FLW. They also proposed the original HMM-based technique for tree sets comparison which is improved in the current work.
2. [14] which is a work published in 2018 where the authors proposed a learning version of the *tree Edit* distance and experimented it on the dataset SSTB2.

It is important to mention that the dataset SSTB2 was more recently experimented in 2020 for tree classification in [34], but we performed no comparison with that work because the authors only randomly selected 100 trees for each class during their experiments. The technique proposed in the current paper:

1. Is based on the HMMs designed in [38] for comparing two tree sets. It consequently inherits the main assets of that technique [14].
2. Unlike [38], it associates a descriptor vector $\overrightarrow{T}\in\mathbb{R}^\alpha$ whose components are interpretable with every tree set T such that all the operations that are applicable to vectors in $\mathbb{R}^\alpha$ are now also applicable to tree sets.

[13] Cf. Section 5.3 of [38]
[14] Cf. Section 5.4 of [38]

3. Performs a finer characterization of a tree set than [38]. Indeed, a tree set was characterized in [38] by its associated HMMs themselves. In the current paper, a tree set is rather characterized by meta-information related to the overall behavior of these HMMs after a sufficiently long time.
4. Exhibits good classification performances (78% for FL and 90.75% for FLW) with the default sets $\overline{P}_L$ and $\overline{P}_{LW}$ of targeted nodes properties, unlike [38] where lower performances were obtained (44.25% for FL and 36% for FLW) in identical conditions.
5. Shows an excellent classification accuracy of 99.75% for FL and for FLW when the suitable sets P_L and P_{LW} of targeted nodes properties are selected. This performance is quite identical to the 100% obtained in [38].
6. Exhibits a good classification accuracy (87.22%) on the real world tree dataset SSTB2, this is 20.52% better than the 66.7% obtained in [14] for this same dataset.
7. Is capable to correctly cluster 54.25%, 98.75% and 75.53% of the instances of FL, FLW and SSTB2 respectively. No clustering was performed in [14, 38].

VI CONCLUSION

The goal of this paper was to improve the HMM-based technique recently proposed in [38] for comparing two tree sets. Given a tree set T and a set $P = \{p_1, \ldots, p_m\}$ of targeted node properties, the principle of the *DFS* algorithm is used to associate $|P|$ HMMs with T, each HMM $\lambda_i(T)$ learning how much the nodes of the trees in T verify property p_i [38]. The resulting models are finally compared to derive a distance between the two sets of trees. The technique proposed in [38] overcame the main limitations of the other existing techniques developed before its publication. Unfortunately, it did not allow the use of non metric-dependent classifiers which absolutely require descriptor vectors as inputs.

In order to derive the descriptor vector $\overrightarrow{T}$ associated with a tree set T in the current paper, we capture the behavior of its associated HMMs by evaluating the overall proportion of time spent by every HMM observing each symbol o after a sufficiently long time, irrespective of the state from which o is observed. The resulting proportions are then sequentially saved as the components of the descriptor vector. The comparison between two tree sets is finally realized by comparing their associated descriptor vectors. Classification experiments carried out on the databases FL, FLW and SSTB respectively showed best accuracies of 99.75%, 99.75% and 87.22%. These performances are respectively 40.75% and 20.52% better than the widespread *tree Edit* distance respectively for FLW and SSTB2. Additional clustering experiments exhibited 54.25%, 98.75% and 75.53% of correctly clustered instances respectively for FL, FLW and SSTB2. These performances illustrate the accuracy of the proposed approach in classification and clustering.

The following perspectives can be explored by future work:

1. The implementation of a modified version of the proposed approach which considers the user-defined importance granted to each node property.
2. The design of manual and automatic techniques for discovering the optimal set of targeted node properties that the best describes the trees in each tree set.
3. The extension of the proposed approach to graph comparison.
4. The reduction of the time cost through the implementation of a parallel computation scheme of the proposed descriptor vectors. This can be achieved by using parallel versions of the *Baum-Welch* algorithm executed on a cluster of computers following [50] or on a *Field-Programmable Gate Array* (FPGA) chip following [51].

5. The visualization of the elements of a tree set T as points scattered in the 2D/3D space using techniques like *Multidimensional Scaling* (MDS) [52]. MDS is a mathematical method which allows easier analysis of data by performing some sort of dimensionality reduction to map high-dimensional data into a lower-dimensional space in such a way that the distances between low-dimensional objects resemble the original similarity information in the high-dimensional space [15]. If each tree $t \in T$ is considered as the singleton $\{t\}$, the proposed approach enables to generate the descriptor vector $\overrightarrow{t}$ associated with $\{t\}$. The tree set T will consequently be viewed as the collection $\tilde{T} = \{\overrightarrow{t} \mid t \in T\}$ of high-dimensional descriptor vectors. MDS to the 2D/3D space can finally be used for visualizing the vectors in $\tilde{T}$ as points scattered in the 2D/3D space.

VII BIOGRAPHY

Sylvain iloga obtained his PhD in Computer Science in January 2018 at the University of Yaoundé 1 in Cameroon. Since January 2010, he has been a teacher in the department of Computer Science of the High Teachers' Training College of Maroua in Cameroon. Between September 2017 and August 2019, he completed a research and teaching internship at the Department of Electronic Engineering and Industrial Computing of the IUT of Cergy-Pontoise at Neuville University, in France. In May 2018, he was promoted to the rank of Lecturer in Cameroon. Subsequently in January 2019, he obtained his qualification for the functions of Lecturer in France, section 27 (Computer Science). His research focuses on machine learning using hidden Markov models, sequential patterns mining, the design of taxonomies for hierarchical classification, the implementation of reconfigurable architectures based on the FPGA technology.

REFERENCES

[1] Gabriel Valiente. An efficient bottom-up distance between trees. In *spire*, pages 212–219, 2001.

[2] Philip Bille. Tree edit distance, alignment distance and inclusion. Technical report, Citeseer, 2003.

[3] Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3):422–433, 1979.

[4] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[5] Kaizhong Zhang, Rick Statman, and Dennis Shasha. On the editing distance between unordered labeled trees. *Information processing letters*, 42(3):133–139, 1992.

[6] Weimin Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40(2):135–158, 2001.

[7] Hélène Touzet. Comparing similar ordered trees in linear-time. *Journal of Discrete Algorithms*, 5(4):696–705, 2007.

[15] See Section 3 of [52]

[8] Erik D Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms (TALG)*, 6(1):2, 2009.

[9] Mateusz Pawlik and Nikolaus Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173, 2016.

[10] Stefan Schwarz, Mateusz Pawlik, and Nikolaus Augsten. A new perspective on the tree edit distance. In *International Conference on Similarity Search and Applications*, pages 156–170. Springer, 2017.

[11] Thorsten Richter. *A new measure of the distance between ordered trees and its applications*. Inst. für Informatik, 1997.

[12] Aïda Ouangraoua, Pascal Ferraro, Laurent Tichit, and Serge Dulucq. Local similarity between quotiented ordered trees. *Journal of Discrete Algorithms*, 5(1):23–35, 2007.

[13] Raghavendra Sridharamurthy, Bin Masood Talha, Kamakshidasan Adhitya, and Natarajan Vijay. Edit distance between merge trees. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 1–14, 2018.

[14] Benjamin Paaßen, Claudio Gallicchio, Alessio Micheli, and Barbara Hammer. Tree edit distance learning via adaptive symbol embeddings. In *International Conference on Machine Learning*, pages 3976–3985. PMLR, 2018.

[15] Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees—an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[16] Jesper Jansson and Andrzej Lingas. A fast algorithm for optimal alignment between similar ordered trees. In *Annual Symposium on Combinatorial Pattern Matching*, pages 232–240. Springer, 2001.

[17] Pekka Kilpeläinen et al. Tree matching problems with applications to structured text databases. 1992.

[18] Laurent Alonso and René Schott. On the tree inclusion problem. In *International Symposium on Mathematical Foundations of Computer Science*, pages 211–221. Springer, 1993.

[19] Pekka Kilpeläinen and Heikki Mannila. Ordered and unordered tree inclusion. *SIAM Journal on Computing*, 24(2):340–356, 1995.

[20] Thorsten Richter. A new algorithm for the ordered tree inclusion problem. In *Annual Symposium on Combinatorial Pattern Matching*, pages 150–166. Springer, 1997.

[21] Weimin Chen. More efficient algorithm for ordered tree inclusion. *Journal of Algorithms*, 26(2):370–385, 1998.

[22] Christoph M Hoffmann and Michael J O'Donnell. Pattern matching in trees. *Journal of the ACM*, 29(1):68–95, 1982.

[23] S Rao Kosaraju. Efficient tree pattern matching. In *30th Annual Symposium on Foundations of Computer Science*, pages 178–183. IEEE, 1989.

[24] Moshe Dubiner, Zvi Galil, and Edith Magen. Faster tree pattern matching. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 145–150. IEEE, 1990.

[25] RAMAKRISHNAN Ramesh and IV Ramakrishnan. Nonlinear pattern matching in trees. *Journal of the ACM (JACM)*, 39(2):295–316, 1992.

[26] KZ Zhang, Dennis Shasha, and Jason Tsong-Li Wang. Approximate tree matching in the presence of variable length don' t cares. *Journal of Algorithms*, 16(1):33–66, 1994.

[27] Tyng-Luh Liu and Davi Geiger. Approximate tree matching and shape similarity. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 456–462. IEEE, 1999.

[28] Martin Farach and Mikkel Thorup. Fast comparison of evolutionary trees. *Information and Computation*, 123(1):29–37, 1995.

[29] Amihood Amir and Dmitry Keselman. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. *SIAM Journal on Computing*, 26(6):1656–1669, 1997.

[30] Sanjeev Khanna, Rajeev Motwani, and F Frances Yao. *Approximation algorithms for the largest common subtree problem*. Citeseer, 1995.

[31] Tatsuya Akutsu and Magnús M Halldórsson. On the approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233(1-2):33–50, 2000.

[32] Arvind Gupta and Naomi Nishimura. Finding largest subtrees and smallest supertrees. *Algorithmica*, 21(2):183–210, 1998.

[33] Naomi Nishimura, Prabhakar Ragde, and Dimitrios M Thilikos. Finding smallest supertrees under minor containment. *International Journal of Foundations of Computer Science*, 11(03):445–465, 2000.

[34] Hikaru Shindo, Masaaki Nishino, Yasuaki Kobayashi, and Akihiro Yamamoto. Metric learning for ordered labeled trees with pq-grams. *arXiv preprint arXiv:2003.03960*, 2020.

[35] Virendrakumar C Bhavsar, Harold Boley, and Lu Yang. A weighted-tree similarity algorithm for multi-agent systems in e-business environments. *Computational Intelligence*, 20(4):584–602, 2004.

[36] K Sasirekha and P Baby. Agglomerative hierarchical clustering algorithm-a. *International Journal of Scientific and Research Publications*, 83:83, 2013.

[37] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. Cluster analysis: basic concepts and algorithms. *Introduction to data mining*, 8:487–568, 2006.

[38] Sylvain Iloga. Customizable hmm-based measures to accurately compare tree sets. *Pattern Analysis and Applications*, pages 1–23, 2021.

[39] To-Yat Cheung. Graph traversal techniques and the maximum flow problem in distributed computation. *IEEE Transactions on Software Engineering*, (4):504–512, 1983.

[40] Mariette Awad and Rahul Khanna. Support vector machines for classification. In *Efficient Learning Machines*, pages 39–66. Springer, 2015.

[41] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[42] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[43] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[44] Jizhou Kang. Stat 243: Stochastic process. https://bookdown.org/jkang37/stochastic-process-lecture-notes/lecture09.html, 2021.

[45] Sylvain Iloga, Alexandre Bordat, Julien Le Kernec, and Olivier Romain. Human activity recognition based on acceleration data from smartphones using hmms. *IEEE Access*, 9:139336–139351, 2021.

[46] Witten Ian H. and Frank Eibe. Data mining: Practical machine learning tools and techniques. http://weka.sourceforge.net/, 2005.

[47] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[48] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[49] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. Parsing With Compositional Vector Grammars. In *EMNLP*. 2013.

[50] J. B. Bogneh Noussi, M. Tchoupé Tchendji, and Sylvain Iloga. Parallel hmm-based similarity between finite sets of histograms. http://cri-info.cm/?page_id=148, 2019.

[51] A. López-López A. Espinosa-Manzo and M. O. Arias-Estrada. Implementing hidden markov models in a hardware architecture. In *Proceedings of the International Meeting of Computer Science (ENC'01), Aguascalientes, Mexico*, volume II, pages 1007–1016, 2001.

[52] Robertas Damaševičius. Analysis of components for generalization using multidimensional scaling. *Fundamenta Informaticae*, 91(3-4):507–522, 2009.