Extended Subtree: A New Similarity Function for Tree Structured Data

Ali Shahbazi, *Student Member, IEEE* and James Miller, *Member, IEEE*

Abstract—Although several distance or similarity functions for trees have been introduced, their performance is not always satisfactory in many applications, ranging from document clustering to natural language processing. This research proposes a new similarity function for trees, namely Extended Subtree (EST), where a new subtree mapping is proposed. EST generalizes the edit base distances by providing new rules for subtree mapping. Further, the new approach seeks to resolve the problems and limitations of previous approaches. Extensive evaluation frameworks are developed to evaluate the performance of the new approach against previous proposals. Clustering and classification case studies utilizing three real-world and one synthetic labeled data sets are performed to provide an unbiased evaluation where different distance functions are investigated. The experimental results demonstrate the superior performance of the proposed distance function. In addition, an empirical runtime analysis demonstrates that the new approach is one of the best tree distance functions in terms of runtime efficiency.

Index Terms—Tree classification, tree clustering, tree comparison, tree distance function, tree similarity, tree structured data

1 Introduction

The extensive application of tree structured data in today's information technology is obvious. Trees can model many information systems like XML and HTML. User behavior in a website (visited pages) [1], [2], [3], proteins, and DNA can be modeled with a tree. Moreover, programming language compilers parse the code into a tree as a first step. Consequently, in many applications involving tree structured data, tree comparison is required. Tree comparison is performed by tree distance/similarity functions. The applications includes document clustering [4], natural language processing [5], cross browser compatibility [6], and automatic web testing [7].

Several tree comparison approaches [8], [9], [10], [11] have been already introduced to address this domain. Edit base distances [10] are a well-known family of tree distances based on mapping and edit operations. They have three major drawbacks with respect to their mapping rules. First, order-preserving rules may prevent mapping between similar nodes, resulting in situations where similar nodes may not contribute towards the overall tree similarity score based solely upon their position. Second, according to the one-to-one condition, any node in a tree can be mapped into only one node in another tree leading to inappropriate mappings with respect to similarity. That is, repeated nodes or structures of mapped nodes have no effect on similarity and they are counted as dissimilar nodes. Finally, edit based distances work based upon mapping individual nodes, not tree structures. This implies that every mapped pair of nodes is independent of all the other nodes. However, a group of mapped nodes should have a stronger emphasis on the similarity of trees when they form an identical subtree. That is, an identical subtree represents a similar substructure between trees, whereas disjoint mapped nodes indicate no similar structure between the two trees. More details of these drawbacks along with illustrative examples are presented in Section 4.1.

In this research, we propose a new similarity function with respect to tree structured data, namely Extended Subtree (EST). The new similarity function avoids these problems by preserving the structure of the trees. That is, mapping subtrees rather than nodes is utilized by new mapping rules. The motivation of proposing EST is to enhance the edit base mappings, provided in Section 3.1, by generalizing the one-to-one and order preserving mapping rules. Consequently, EST introduces new rules for subtree mapping. This new approach seeks to resolve the problems and limitations of edit based approaches (this is detailed in Section 4.1 with illustrative examples).

To evaluate the performance of the proposed similarity function against previous researches, an extensive experimental study is performed. The experimental evaluation frameworks include clustering and classification frameworks. The distance functions provide the core functionality for clustering and classification applications. In addition, four distinct data sets (three real and one synthetic) are utilized to perform the evaluation.

In general, this research's contributions can be summarized as:

- Introducing a novel similarity function to compare tree structured data by defining a new set of mapping rules where subtrees are mapped rather than nodes.
- Further, the new approach resolves the limitations of the previous distance functions.

- *The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada. E-mail: ali.shahbazi@ualberta.ca, jm@ece.ualberta.ca.*

Manuscript received 1 May 2012; revised 2 Oct. 2012; accepted 10 Mar. 2013; date of publication 26 Mar. 2013; date of current version 18 Mar. 2014.
Recommended for acceptance by Y. Tao.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2013.53

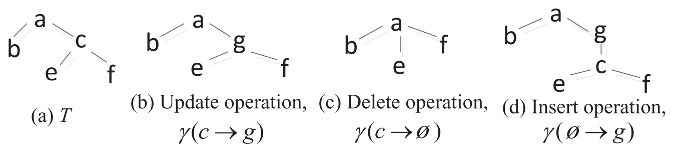

(a) T
(b) Update operation, $\gamma(c \to g)$
(c) Delete operation, $\gamma(c \to \emptyset)$
(d) Insert operation, $\gamma(\emptyset \to g)$

Fig. 1. Three edit operations, "delete", "insert", and "update".

- Designing extensive evaluation frameworks using k-medoid [12], KNN (K Nearest Neighbor) [13], and support vector machine (SVM) [14] along with four different data sets to perform an unbiased evaluation. That is, we believe applying one machine learning technique on a single data set might lead to a biased evaluation; and hence, it is not considered adequate to prove the effectiveness of an approach. This extensive evaluation framework is one of the advantages of this research over previous researches such as [1], [3], [4], [15], and [16].
- Superior results of EST against previous approaches in most of the clustering and classification case studies.
- Empirical runtime analysis of the new approach as well as current approaches where runtime efficiency of EST is demonstrated.

The rest of this paper is organized as follows. Section 2 explains notations and definitions regarding trees. An in-depth description of the current approaches to tree distance calculations is presented in Section 3. Section 4 proposes the EST similarity function along with its algorithms and runtime analysis. Section 5 introduces experimental frameworks to evaluate the proposed approach against previous distance functions. The performance of EST with illustrative results as well as empirical runtime analysis is demonstrated in Section 6. Finally, conclusions are drawn in Section 7.

2 Tree Notation and Definitions

The following notation and assumptions are provided with respect to trees to simplify the discussion in this paper. In this paper, trees are referring to rooted, ordered, and labeled trees unless otherwise stated. A rooted tree is a tree with a single root node. A tree is ordered if right-left order amongst sibling nodes in the tree are important. Finally, a labeled tree represents a tree where each node has an assigned label.

A tree is denoted as T and $|T|$ indicates the size of a tree in terms of the number of nodes/vertices. Multiple trees are differentiated by a top index as T^p and T^q. t_i represents the ith node of T numbered in a post-order format. In case of multiple trees, again a top index is utilized to distinguish between trees. For instance, t_i^p and t_i^q. In this paper, $V(T)$ defines a set of vertices/nodes of T where $V(T) = \{t_i\}_{i=1}^{|T|}$. The depth of a tree is denoted by $depth(T)$ which is defined as the length of the path from the root to the deepest node in the tree. $depth(t_i)$ indicates the length of the path from the root to t_i. $leaves(T)$ indicates the number of leaves in T where a leaf node is the node without children. deg (t_i) represents the degree of node t_i which is equal to the number of t_i's children. Accordingly, $deg\,(T)$ represents the degree of T, which is the maximum number of children of any node in the tree. A subtree is a tree which is part of a larger tree. Accordingly, T_i denotes a subtree of T rooted at t_i. If T^p is a subtree of T^q, we indicate it as $T^p \subset T^q$.

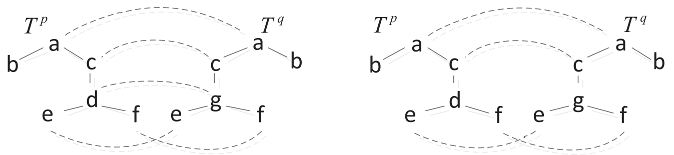


Fig. 2. Optimal mappings between trees for TED and IST.

Finally, distance and similarity between T^p and T^q are presented as $D(T^p, T^q)$ and $S(T^p, T^q)$, respectively. Similarly, normalized values are indicated by D^* and S^* where we have

$$S^*(T^p, T^q) = 1 - D^*(T^p, T^q). \tag{1}$$

3 Current Approaches

A variety of different tree distance functions have been proposed. In this section, we survey these approaches and present a summary of each one.

3.1 Edit Based Distances

Edit based distances [10] work is based on three edit operations (γ) including "delete", "insert", and "update" [17] (Fig. 1). Each operation has an associated cost (W_{delete}, W_{insert}, W_{update}). Based on the introduced edit operations, each tree can be converted into another tree according to a set of rules that are different for each distance function. Further, mappings were introduced in [18] to describe how a sequence of edit operations converts a tree into another tree [19], namely T^p and T^q respectively. Fig. 2 represents a sample T^p and T^q along with a few mappings where each mapping represents an optimal mapping associated with a tree distance approach. A mapping is a set of ordered integers such as (i_p, i_q) where i_p and i_q are the index of the nodes (numbered in post-order format) from tree T^p and T^q, respectively. This means that node t_{ip}^p is mapped into node t_{iq}^q. The following conditions must be satisfied for all (i_p, i_q), $(j_p, j_q) \in M$ [19]:

- One-to-one condition: $i_p = j_p$ if and only if $i_q = j_q$. This condition implies that one node from T^p cannot be mapped into two nodes from T^q.
- Sibling order preservation condition: $i_p > j_p$ if and only if $i_q > j_q$.
- Ancestor order preservation condition: t_{ip}^p is an ancestor of t_{jp}^p if and only if t_{iq}^q is an ancestor of t_{jq}^q.

$D(T^p, T^q)$ is equal to the cost of the edit operations required to convert T^p into T^q. Assuming the cost of each edit operation as one, the $D(T^p, T^q)$ is bounded between zero and $|T^p| + |T^q|$. Accordingly, it can be normalized between zero and one as:

$$D^*(T^p, T^q) = \frac{D(T^p, T^q)}{|T^p| + |T^q|}. \tag{2}$$

3.1.1 Tree Edit Distance (TED)

TED [17], [18], [20] is a well-known edit based distance function that measures the minimum cost of a sequence of edit operations between two trees. Since its introduction by Tai [18], several algorithms have been introduced for computing the optimal TED between two trees. This research follows the dynamic programming presented by Zhang and Shasha [17]. The computational order for this algorithm is $D_{TED}(T^p, T^q) \in O(|T^p||T^q|Min(depth(T^p), leaves(T^p))\ Min(depth(T^q), leaves(T^q)))$ [17] where $O(.)$ represents the runtime order. The TED mapping needs only to satisfy the mapping's conditions presented in previous section. The mapping demonstrated in Fig. 2a indicates an optimal mapping to calculate the TED. According to this mapping $D_{TED}(T^p, T^q) = 3$, since we have only one update operation $(\gamma(d \rightarrow g))$, one insert operation $(\gamma(\emptyset \rightarrow b))$, and one delete operation $(\gamma(b \rightarrow \emptyset))$.

3.1.2 Isolated Subtree (IST) Distance

The IST distance is introduced by Tanaka [21], it maps the disjoint subtrees of T^p to the similar disjoint subtrees of T^q. Tanaka and Tanaka [21] argued that such a mapping is more meaningful since it preserves the structure of the trees. The IST mapping is a TED mapping where disjoint subtrees are mapped to similar disjoint subtrees under the restriction of the structure preserving mapping [21]. Fig. 2b demonstrates the optimal IST mapping between T^p and T^q. In this sample, $D_{\mathrm{IST}}(T^p, T^q) = 0 + 2 + 2 = 4$. Tanaka and Tanaka [21] provided an algorithm to compute the optimal IST distance with the runtime complexity of $O(|T^p| \times\ |T^q| \times Min(leaves(T^p), leaves(T^q)))$ [21], [22]. Later, Zhang [23] provided an algorithm to calculate IST distance with runtime complexity of $O(|T^p| \times |T^q|)$.

In addition to TED and IST, there are other distance functions including alignment [24], top-down [25], and bottom-up distance [22]. Their objective is to simplify the calculations; however, they produce lower quality solutions than TED.

3.2 Multisets Distance

Recently, Müller-Molina et al. [11] have introduced a tree distance metric based on multisets. Multisets are sets that allow repeated elements, where T^p and T^q are converted into multisets, M^p and M^q. M^p and M^q contain all the complete subtrees of the corresponding trees. A complete subtree is defined as a subtree that: if t_i is a node in a complete subtree, all of t_i's children are in the subtree. In addition, $V(T^p)$ and $V(T^q)$ are utilized along with M^p and M^q to calculate $D(T^p, T^q)$ as:

$$\begin{aligned} D_{multiset}(T^p, T^q) = ((|M^p \cup M^q| - |M^p \cap M^q|) \\ + (|V(T^p) \cup V(T^q)| - |V(T^p) \cap V(T^q)|))/2. \end{aligned} \tag{3}$$

Müller-Molina et al. [11] presented no approach for normalization. However, the normalized distance can be calculated using (2) since $D_{multiset}(T^p, T^q)$ is bounded between 0 and $|T^p| + |T^q|$. An algorithm with runtime complexity of $O(|T^p| \times |T^q|^2)$ is presented in [11] to compute $D_{multiset}(T^p, T^q)$.

3.3 Path Distance

Path distance [9] considers paths as a tree's building blocks. Each tree is converted into a multiset of paths such as "/a/c/d" which describes a path in T^p in Fig. 2a. Different approaches exist to extract paths from a tree. One possible approach is that all paths start from a root node to t_i. Any node to any possible node is another approach where a path to t_i can start from any ancestor of t_i or even t_i. The later approach includes all the possible paths in the tree. In this research, we follow the second approach for path extraction. Given T^p and T^q, M^p and M^q are the multisets which contain all the paths in T^p and T^q, respectively. $S_{path}(T^p, T^q)$ can be simply calculated as $|M^p \cap M^q|$. Since $S_{path}(T^p, T^q)$ is bounded between zero and $Max(|T^p|, |T^q|)$, $S^*_{path}(T^p, T^q) = \frac{|M^p \cap M^q|}{Max(|T^p|, |T^q|)}$.

3.4 Entropy Distance

Connor et al. [8] utilized information theory, Shannon's entropy, to calculate a bounded, between zero and one, distance function between two trees. Similar to the path distance metric, the M^p and M^q multisets are generated which contain all the possible paths in T^p and T^q, respectively. Then, Shannon's entropy equation and complexity theory are used to calculate the information distance. Finally, Connor et al. [8] concluded the distance as:

$$D_{Entropy}(T^p, T^q) = \frac{C(M^p \uplus M^q)}{\sqrt{C(M^p) \times C(M^q)}} - 1, \tag{4}$$

where $\uplus$ represents the union of two multisets; and $C(M)$ denotes complexity of a multiset defined as [8]

$$C(M) = b^{H_b(M)} = b^{-\Sigma_i p(m_i) \log_b p(m_i)} = \prod_i p(m_i)^{-p(m_i)}, \tag{5}$$

where b is a constant number, $H_b(M)$ represents the entropy of M in base b, and m_i denotes a member of M where i represents all the distinct members of M. Finally, $p(m_i)$ denotes the probability of m_i in M which is equal to the number of m_i repetitions over $|M|$. The authors did not provide the order of runtime complexity of the algorithm.

In addition to the discussed approaches, Lu [26] introduced node splitting and merging. Further, Helmer [27] utilized Kolmogorov complexity which provides a new class of distances for measuring similarity relations between sequences [28]. The main advantage of this approach is its linear runtime complexity which is reported [27] as $O(|T^p| + |T^q|)$. Finally, Yang et al. [29] introduced a distance measure between two trees based on a numeric vector representation of trees. They prove that this distance, $D_{binary}(T^p, T^q)$, is a lower bound for $D_{TED}(T^p, T^q)$ given by $D_{binary}(T^p, T^q) \leq 5 \times D_{TED}(T^p, T^q)$ and hence, it has lower quality compared to TED. However, it has a linear runtime complexity given by $O(|T^p| + |T^q|)$ which outperforms TED in this respect.

The proposed distance function's (EST) performance is evaluated against TED, IST, Entropy, Multisets, and Path distances as no compelling evidence exists that any other superior techniques exists and no comprehensive comparison of these techniques appears in the literature. However, it should be noted that approaches such as Kolmogorov

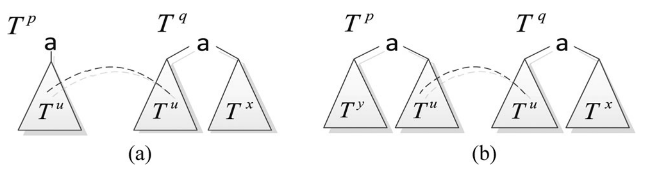


Fig. 3. Samples of T^p and T^q utilized to problems regarding mapping conditions in edit based distances.

complexity [27] and Binary distance [29] have linear computational complexity; and hence, have a superior runtime to those used in the experiments.

4 Proposed Tree Similarity Function: Extended Subtree

In this section, we propose a new similarity function, namely EST, to compare trees. The new function seeks to resolve many of the issues which will be discussed in the following section. Further, a computational algorithm as well as its runtime complexity is presented.

4.1 Motivation

In this section, we justify the need to propose a new tree comparison approach by discussing situations where previous approaches have poor performance. Note that the aim of the new approach in not runtime complexity reduction as presented in [29], [30]. Although runtime complexity is an important issue in practical applications, we focus on proposing a new approach that better represents the similarity or distance between tree-structured data. This leads to an enhancement in applications where a tree distance function is utilized.

A variety of tree comparison approaches are introduced in the previous section. Each approach has advantages and disadvantages in terms of the distance/similarity score. Based upon an empirical investigation, we found situations where the previous approaches do not give an appropriate similarity/distance score. In the following, these cases are analyzed with illustrative examples where all discussions are in terms of a normalized similarity score, $S^*(T^p, T^q)$. $S^*(T^p, T^q) = 1$ means that the trees are identical; while $S^*(T^p, T^q) = 0$ means that the trees are totally distinct.

All of the five edit based tree distance approaches follow the mapping rules presented in Section 3.1, namely one-to-one and order preserving conditions. According to the one-to-one condition, any node in T^p can only be mapped to one node in T^q. Now consider Fig. 3a where $T^u \subset T^p$ and $T^u, T^x \subset T^q$. Also assume that $|T^u|, |T^x| \gg 1$, so the cost of the root nodes in T^p and T^q have negligible impact on the distance calculation. Considering $|T^u| = |T^x|$ in Fig. 3a leads to $S^*(T^p, T^q) \simeq 0.667$ with respect to all five edit based approaches. There is a problem in this similarity score: no matter whether T^u and T^x are identical or totally different, $S^*(T^p, T^q)$ remains 0.667. The one-to-one mapping condition enforces that T^x cannot be mapped to $T^u \subset T^p$, since $T^u \subset T^p$ is already mapped to $T^u \subset T^p$. Moreover, according to the order preserving conditions, a node in T^p can be mapped to one node in T^q, if the ordering is preserved with other mappings. This is how edit based distances differentiate between ordered and unordered trees. This rule seems

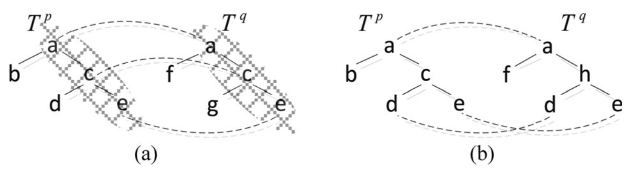


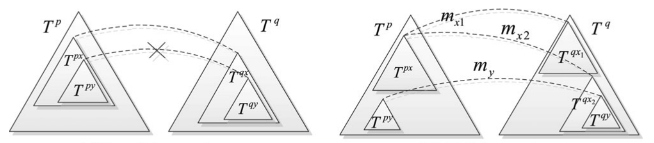
Fig. 4. Samples of isolated subtree mappings where (a) the mapped nodes form a subtree as denoted by the hatches; and (b) the mapped nodes are separate nodes.

less than ideal in a number of situations. To clarify this discussion, consider Fig. 3b, where $T^u, T^y \subset T^p$ and $T^u, T^x \subset T^q$; again, assume that $|T^u|, |T^x|, |T^y| \gg 1$. Considering $|T^u| = |T^x| = |T^y|$ in Fig. 3b leads to $S^*(T^p, T^q) \simeq 0.5$ with respect to all edit based approaches. The problem in this case is that whether T^x and T^y are identical or totally different, the similarity score remains at 0.5. This means that when considering T^x and T^y as identical, they cannot be mapped together due to the order preserving conditions. Please note that considering T^x and T^y as identical does not lead to $T^p = T^q$, since T^p and T^q are ordered trees. Accordingly, we are not discussing that by mapping T^y to T^x, the similarity score would be one. What we are discussing is that if $T^x = T^y, 0.5 < S^*(T^p, T^q) < 1$ better represents the similarity between these trees. According to these discussions, we introduce a new set of mapping conditions in the next section.

Further, we observed that m (a constant number) similar nodes between T^p and T^q have a stronger emphasis on the similarity of T^p and T^q when they form an identical subtree mapping between T^p and T^q (Fig. 4a), compared to disjoint nodes as illustrated in Fig. 4b. That is, an identical subtree represents a similar substructure between T^p and T^q, whereas m disjoint mapped nodes indicate no similar structure between the two trees. However, edit based approaches, in particular the IST distance, are unable to model this. That is, in the IST distance, m mapped disjoint nodes have the same similarity as m nodes forming a subtree. Fig.4 represents two IST mappings where $S^*(T^p, T^q) = 0.6$ in both cases. However, we believe that T^p and T^q presented in Fig. 4a are more similar than the trees presented in Fig. 4b, since Fig. 4a contains a similar subtree as denoted by the hatches.

Path [9] and entropy [8] distances consider paths as a tree's building blocks as their basic assumption; that is, they convert a tree into a multiset of paths and then compare the trees by comparing the multisets of paths. This assumption is not in accordance with the nature of tree-structured data. If we could convert a tree into a multiset of paths, there would have been no reason to present the data initially as a tree. Further, the entropy approach produces some strange results. Assuming the trees presented in Fig. 3a with the aforementioned conditions regarding T^u and T^x, the entropy approach yields $S^*(T^p, T^q) \simeq 1$ where $T^u = T^x$. Obviously, this result is unsatisfactory as T^p and T^q are not identical.

The binary [29] and Fourier [30] distances assume TED as an ideal distance approach and approximate TED while reducing the runtime complexity. Fourier distance converts a tree to a signal in the frequency domain. This approach does not make sense at all with respect to tree comparisons. The poor performance of Fourier distance, presented in [9], verifies that it is not an appropriate tree comparison



(a) Invalid mapping (b) Valid mapping

Fig. 5. Extended Subtree mapping where (a) indicates invalid mappings, and (b) represents valid mappings.

approach. The bottom-up approach [22] puts more value on bottom nodes rather than top nodes, since it matches the bottom nodes first. Therefore, this approach is useless in most of the situations where nodes have equal weights or where top nodes have larger weights. Based on our empirical investigation, the multiset approach [11] behavior is similar to the bottom-up approach in terms of putting more value on bottom nodes; that is, every subtree defined in this approach contains leaves of the tree. Finally, the NCD approach [27] does not seem an appropriate distance metric, since it converts the tree into plain text where each node's label is converted to text. Just as an example to demonstrate a disadvantage of this approach, assume that different nodes are labeled with different numbers in a tree like 2, 111, and 1111. All the three labels are different, but since the labels are converted into plain text, 111 and 1111 are considered similar in the compression process utilized in NCD. Further, since an optimal compressor does not exist, a real world compressor is utilized which does not yield optimal NCD scores.

As a conclusion, the main motivation for proposing a new tree similarity approach is introducing an approach which resolves the discussed problems and removes the limitations of the previous approaches. In addition, the new approach must enhance the applications where a tree distance function is utilized.

4.2 Extended Subtree Similarity

Given T^p and T^q, the proposed EST preserves the structure of the trees by mapping subtrees of T^p to similar subtrees of T^q. Although it might seem similar to the IST, it is fundamentally different since EST's mappings are not in accordance with the mapping conditions provided in Section 3.1. That is, EST generalizes the edit base distances and mappings. According to the discussions in the previous section, given T^{px} and T^{qx} as two mapped subtrees in T^p and T^q with m_x as the name of this mapping, we introduce the rules of the new approach's mapping as:

Rule 1: EST's mapping is a subtree mapping which means that not only single nodes can be mapped together, but also identical subtrees can be mapped together (unlike IST). Using subtree mapping, we can increase the significance of larger subtrees, since they are considered more important than single nodes in accordance with the discussion in the previous section.

Rule 2: No common subtrees of T^{px} and T^{qx} are allowed to be mapped together, as indicated in Fig. 5a, this is defined as an invalid mapping. When two subtrees of T^{px} and T^{qx} are already mapped, all the sub structures of T^{px} and T^{qx} can be mapped together as T^{px} and T^{qx} are identical. Since we are interested in larger mapped subtrees, mapped subtrees of T^{px} and T^{qx} have no use, so we categorize them as invalid mappings.

Rule 3: One-to-many condition: A subtree of T^p can be mapped into several subtrees of T^q and vice versa. The intuition of this rule is with respect to Fig. 3a where the disadvantages of the one-to-one condition are investigated. As indicated in Fig. 5b, T^{px} is mapped to T^{qx1} and T^{qx2} concurrently. Further, T^{qy} is mapped into T^{py} where T^{qy} is a subtree of T^{qx2} which is already mapped.

Rule 4: m_x is weighted as $W(m_x) = (W(T^{px}) + W(T^{qx}))/2$ where $W(T^{px})$ and $W(T^{qx})$ are the weights of subtrees in the mapping. $W(T^{px})$ (and similarly for $W(T^{qx})$) is calculated as:

$$W(T^{px}) = \sum_{t_i^{px} \in T^{px}} W\left(t_i^{px}\right), \tag{6}$$

where $W(t_i^{px})$ is the unit scalar, when T^{px} is the largest subtree that t_i^{px} belongs to; and zero otherwise. A node like t_i^{px} might be a member of several subtrees in the mappings as indicated in Fig. 5b. However, it is inappropriate to multiply-count the same node; therefore, nodes are counted as a weight just for the largest subtree that they belong to.

Finally, we can compute $S(T^p, T^q)$ based on all the possible valid mappings as:

$$S(T^p, T^q) = \sqrt[\alpha]{\sum_{m_k \varepsilon M} \beta_k \times W(m_k)^{\alpha}}, \tag{7}$$

where $\alpha, \alpha \geq 1$, is a coefficient to adjust the relation among different sizes of mappings. It amplifies the importance of large subtrees compared to small subtrees or single nodes in accordance with the discussion in the previous section. This similarity function has obvious parallels with the Minkowski distance function [31] which is a popular distance function for higher dimensions of data. $\alpha = 1$ does not amplify the importance of large subtrees compared to small subtrees. As α grows larger, more emphasis is placed on larger subtrees. Further, β_k is a geometrical parameter which reflects the importance of the mapping with respect to the position of T^{pk} and T^{qk} in T^p and T^q, respectively. β_k is the unit scalar, when the root nodes of T^{pk} and T^{qk} have the same depth with respect to T^p and T^q; and it is equal to β (a constant number between zero and one) otherwise; leading to the amplification of the mapping of the same depth regarding subtrees. The selection of α and β values are discussed in Section 5.5.

To normalize the similarity score, we divide it by its higher bound. Since $0 \leq \beta_k \leq 1$, we have $S(T^p, T^q) \leq \sqrt[\alpha]{\sum_{m_k \varepsilon M} W(m_k)^{\alpha}}$. Further, $\sqrt[\alpha]{\sum_{m_k \varepsilon M} W(m_k)^{\alpha}} \leq \sum_{m_k \varepsilon M} W(m_k)$ where $\alpha \geq 1$ and $W(m_k)$ is a positive number. In addition, each node is counted as one in the weight calculation, $\sum_{m_k \varepsilon M} W(m_k) \leq Max(|T^p|, |T^q|)$. As a result, $S(T^p, T^q) \leq Max(|T^p|, |T^q|)$ and the similarity function is normalized as:

$$S^*(T^p, T^q) = \frac{S(T^p, T^q)}{Max(|T^p|, |T^q|)}. \tag{8}$$

In the example provided in Fig. 5b, consider the presented mappings as the only valid mappings. In addition, assume $|T^{px}| = |T^{qx1}| = |T^{qx2}| = 5$ and $|T^{py}| = |T^{qy}| = 2$. Therefore, mapping weights can be computed as $W(m_{x1}) = 5, W(m_{x2}) = 2.5$, and $W(m_y) = 1$. Accordingly, if we consider $\alpha = 2$ and $\beta = 1$, the similarity score is $S(T^p, T^q) = \sqrt{5^2 + 2.5^2 + 1^2} = 5.679$. Consequently, considering $|T^p| = 8$ and $|T^q| = 10$, the normalized similarity score is $S^*(T^p, T^q) = 0.568$.

4.3 Computational Algorithm

Assume $T^p_{i,j}$ represents a subtree of T^p rooted at t^p_i which is mapped to a identical subtree of T^q rooted at t^q_j, namely $T^q_{j,i}$. Accordingly, computing $S(T^p, T^q)$ has four following steps.

Step 1: *Identify all the mappings*: In this step, we find all the possible mappings, valid or invalid (in Step 3, invalid mappings will have a zero weight), and store two lists of nodes for each mapping, one for each subtree. T^p and T^q are the inputs to this step and V^p and V^q are the outputs (inputs for the next step). V^p and V^q are two dimensional matrices where each element is a list of nodes. Accordingly, $V^p[i][j]$ and $V^q[j][i]$ represent the list of nodes of the mapped subtrees of $T^p_{i,j}$ and $T^q_{j,i}$, respectively. The pseudo code represented in Fig. 6 details this step's calculations. The *GetMapping(i, j)* function produces two lists of nodes ($V^p[i][j]$ and $V^q[j][i]$) for a mapping. Its objective is to detect the largest possible mapping. To achieve this objective, we need to find and match the mappings rooted at the children of t^p_i and t^q_j. Since i and j are node indexes in post-order formatting, when computing *GetMapping(i, j)* for nodes t^p_i and t^q_j, the computation is already performed for all the children of t^p_i and t^q_j in advance. Therefore, as indicated in the pseudo code, the *GetMapping(i, j)* function goes through all of the children of t^p_i and t^q_j to use the mapping information among t^p_i's and t^q_j's children to find the largest mapping between t^p_i and t^q_j. t^p_{ia} denotes the ath child of the t^p_i node, where $1 \leq a \leq \deg(t^p_i)$, and *ia* represents the index of the ath child of the t^p_i node. Similarly, t^q_{jb} represents the bth child of the t^q_j node, where $1 \leq b \leq \deg(t^q_j)$ and *jb* represents the index of the bth child of the t^q_j node. In Fig. 6, E is a matrix which indicates how the children of t^p_i and t^q_j are matched. Accordingly, E is used to update $V^p[i][j]$ and $V^q[j][i]$. Since $T^p_{i,j}$ and $T^q_{j,i}$ are identical, $|V^p[i][j]| = |V^q[j][i]|$, so $|V^p[i][j]|$ can be replaced by $|V^q[j][i]|$ in the pseudo code.

Step 2: *Identify each node's largest mapping*: A node in T^p or T^q might belong to several mappings. Considering that we do not want to count one node several times, we determine the largest subtree in the mappings for each node. To compute this step, first, assume two arrays, namely LS^p and LS^q, of size $|T^p|$ and $|T^q|$, respectively. $LS^p[i]$ indicates the largest subtree that t^p_i belongs to by the indexes of root nodes of the mapping, denoted by $LS^p[i]_{mi}$ and $LS^p[i]_{mj}$. As indicated in Fig. 6, filling LS^p and LS^q with appropriate values is the objective of this step. For each mapping, between $T^p_{i,j}$ and $T^q_{j,i}$, we iterate through all the nodes in $V^p[i][j]$ and $V^q[j][i]$ which were computed in the first step. For each node in $V^p[i][j]$, where the index of the node is denoted by $V^p[i][j]_k$ in the pseudo code, we check if the $|V^p[i][j]|$ is larger than the subtree stored in LS^p for that node, and update LS^p accordingly. A similar procedure is repeated for each node in $V^q[j][i]$.

```
Step 1
Begin
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      if $label(t^p_i) == label(t^q_j)$ then
        $GetMapping(i, j)$
      end of if
    end of for
  end of for
Step 2
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      for $k$ = 1 to $|V^p[i][j]|$ do
        $i' \leftarrow V^p[i][j]_k$,  $j' \leftarrow V^q[j][i]_k$
        if $|V^p[i][j]| > |V^p[LS^p[i']_{mi}][LS^p[i']_{mj}]|$ then
          $LS^p[i']_{mi} = i$, $LS^p[i']_{mj} = j$
        end of if
        if $|V^q[j][i]| > |V^q[LS^q[j']_{mj}][LS^q[j']_{mi}]|$ then
          $LS^q[j']_{mi} = i$, $LS^q[j']_{mj} = j$
        end of if
      end of for
    end of for
  end of for
Step 3
  for $i$ = 1 to $|LS^p|$ do
    $W^p[LS^p[i]_{mi}][LS^p[i]_{mj}] + +$
  end of for
  for $j$ = 1 to $|LS^q|$ do
    $W^q[LS^q[j]_{mj}][LS^q[i]_{mi}] + +$
  end of for
Step 4
  for $i$ = 1 to $|T^p|$ do
    for $j$ = 1 to $|T^q|$ do
      $temp = \left(\frac{W^p[i][j] + W^q[j][i]}{2}\right)^\alpha$
      if $depth(t^p_i) \neq depth(t^q_j)$ then
        $temp = temp \times \beta$
      end of if
      $S = S + temp$
    end of for
  end of for
  $S = \sqrt[\alpha]{S}$
End
Step 1, $GetMapping(i, j)$ function
Begin of $GetMapping(i, j)$
  $V^p[i][j] = \{t^p_i\}$
  $V^q[j][i] = \{t^q_j\}$
  for a = 1 to deg($t^p_i$) do
    for b = 1 to deg($t^q_j$) do
      $E[a][b] = Max\begin{cases} E[a-1][b] \\ E[a][b-1] \\ E[a-1][b-1] + |V^p[ia][jb]| \end{cases}$
    end of for
  end of for
  a= deg($t^p_i$)
  b= deg($t^q_j$)
  while $a > 0$ and $b > 0$ then
    if E[a][b] == E[a − 1][b − 1] + $|V^p[ia][jb]|$ then
      $V^p[i][j] = V^p[i][j] \cup V^p[ia][jb]$
      $V^q[j][i] = V^q[j][i] \cup V^q[jb][ia]$
      $a = a - 1$
      $b = b - 1$
    else if E[a][b] == E[a][b − 1] then
      $b = b - 1$
    else
      $a = a - 1$
    end of if
  end of while
End
```

Fig. 6. Pseudo code for the proposed algorithm.

Step 3: *Compute the weight of each subtree*: In this step, we calculate $W(T^p_{i,j})$ and $W(T^q_{j,i})$ for all the subtrees in the mappings. In the pseudo code, they are denoted by $W^p[i][j]$ and $W^q[j][i]$. We go through LS^p and increase the weight of a subtree when it is reported as a largest subtree of a node in LS^p. This procedure is repeated for LS^q as well.

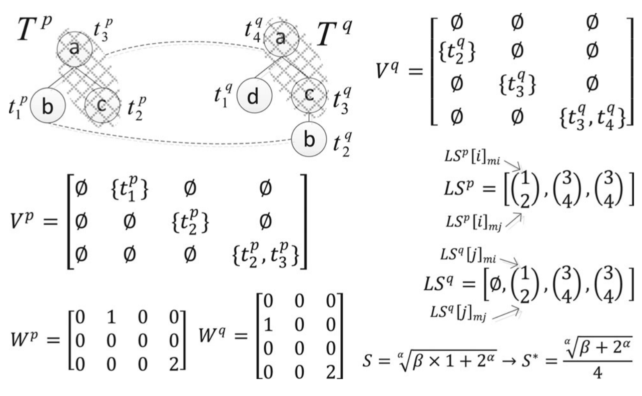

T^p
T^q
t_1^p t_2^p t_3^p
t_1^q t_2^q t_3^q t_4^q
$V^q = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ \{t_2^q\} & \emptyset & \emptyset \\ \emptyset & \{t_3^q\} & \emptyset \\ \emptyset & \emptyset & \{t_3^q, t_4^q\} \end{bmatrix}$
$V^p = \begin{bmatrix} \emptyset & \{t_1^p\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \{t_2^p\} & \emptyset \\ \emptyset & \emptyset & \emptyset & \{t_2^p, t_3^p\} \end{bmatrix}$
$LS^p = \left[\binom{1}{2}, \binom{3}{4}, \binom{3}{4}\right]$
$LS^p[i]_{mi}$ $LS^p[i]_{mj}$
$LS^q = \left[\emptyset, \binom{1}{2}, \binom{3}{4}, \binom{3}{4}\right]$
$LS^q[j]_{mi}$ $LS^q[j]_{mj}$
$W^p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$
$W^q = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$
$S = \sqrt[\alpha]{\beta \times 1 + 2^\alpha} \rightarrow S^* = \frac{\sqrt[\alpha]{\beta + 2^\alpha}}{4}$

Fig. 7. A simple example for the proposed EST algorithm.

Step 4: *Calculate* $S(T^p, T^q)$: Now that we have all the subtree weights (W^p and W^q) available, we can simply calculate $S(T^p, T^q)$ according to (7).

Fig. 7 presents an example which indicates the inputs and outputs of each step. Two simple trees with three and four nodes are presented where the mappings are indicated on the figure. There is two valid mapping, one between nodes b and the other between subtrees of a-c. According to Step 1, V^p and V^q are calculated which indicates all the valid and invalid mappings. The largest subtree, for each node, are calculated using Step 2 and are saved in LS^p and LS^q. For example, the second element of LS^p, $\binom{3}{4}$, indicates the largest subtree that t_2^p is a member of. $\binom{3}{4}$ represents the mapping between subtrees rooted at node 3 of T^p, and node 4 of T^q. In the next step, the weight of each subtree in the mapping is calculated and stored in W^p and W^q. Finally, in Step 4, similarity is calculated from W^p and W^q.

4.4 Runtime Complexity Analysis

In this section, we discuss the order of computational complexity of the EST algorithm. The order of runtime complexity is the summation of the associated complexity into each of the four steps, discussed in the previous section. In accordance with the pseudo code presented in Fig. 6, we have $EST_{step1} \in O(\sum_{i=1}^{|T^p|} \sum_{j=1}^{|T^q|} (GetMapping))$. The *GetMapping* function has a double "for" loop and a "while" loop. Obviously, the double "for" loop is executed $\deg(t_i^p) \times \deg(t_j^q)$ times and the maximum number of executions of the "while" loop is $\deg(t_i^p) + \deg(t_j^q)$. Inside the "while" loop we have two set's union operations which has runtime complexity of $O(Min(|T^p|, |T^q|))$, since the size of each subtree in the mapping cannot be larger than $Min(|T^p|, |T^q|)$. Accordingly, *GetMapping* $\in O(\deg(t_i^p) \times \deg(t_j^q) + (\deg(t_i^p) + \deg(t_j^q)) \times Min(|T^p|, |T^q|))$. Consequently, $EST_{step1} \in O(\sum_{i=1}^{|T^p|} \sum_{j=1}^{|T^q|} (\deg(t_i^p) \times \deg(t_j^q)) + Min(|T^p|, |T^q|) \times \sum_{i=1}^{|T^p|} \sum_{j=1}^{|T^q|} (\deg(t_i^p) + \deg(t_j^q)))$. This result can be simplified to $O(\sum_{i=1}^{|T^p|} \deg(t_i^p) \times \sum_{j=1}^{|T^q|} \deg(t_j^q) + Min(|T^p|, |T^q|) \times (\sum_{i=1}^{|T^p|} \sum_{j=1}^{|T^q|} \deg(t_i^p) + \sum_{i=1}^{|T^p|} \sum_{j=1}^{|T^q|} \deg(t_j^q)))$. Since $\sum_{i=1}^{|T^p|} \deg(t_i^p) = |T^p| - 1 < |T^p|$, we have $EST_{step1} \in O(|T^p| \times |T^q| + 2 \times Min(|T^p|, |T^q|) \times |T^p|) \times |T^q|)$. Since we need to keep the term with highest order, the final runtime of the algorithm's Step 1 is given by $O((|T^p| \times |T^q|) \times Min(|T^p|, |T^q|))$.

TABLE 1
Detailed Information Regarding the Real and Synthetic Data Sets

Data set	Data set size	Average tree size	Average tree depth
CSLOG_2C_WEEK1	8074	8.03	4.40
CSLOG_2C_WEEK2	7407	8.05	4.46
CSLOG_2C_WEEK3	7628	7.98	4.42
SIGMOD_3C	987	39.04	3.76
TREEBANK_2C	160,616	13.08	4.64
TREEBANK_5C	769,172	9.48	3.85
TREEBANK_6C	922,442	12.80	4.52
SYN_2C_CLUSTER	100	58.22	8.48
SYN_3C_CLUSTER	150	58.50	8.44
SYN_5C_CLUSTER	250	58.31	8.51
SYN_8C_CLUSTER	400	57.85	8.45
SYN_20C_CLUSTER	1000	11.93	5.08
SYN_2C_CLASIFY	200	57.32	7.78
SYN_3C_CLASIFY	300	57.62	7.84
SYN_5C_CLASIFY	500	57.31	7.85
SYN_8C_CLASIFY	800	57.20	7.84
SYN_20C_CLASIFY	2000	12.81	4.83

To determine LS^p and LS^q in Step 2, we need to iterate $|T^p| \times |T^q|$ times to cover all the possible mappings. According to the description in Step 2, the complexity of each mapping's iteration is $O(2 \times Min(|T^p|, |T^q|))$ since $V(T_{i,j}^p)$ and $V(T_{j,i}^q)$ are bounded between zero and $Min(|T^p|, |T^q|)$. Therefore, $EST_{step2} \in O(|T^p| \times |T^q| \times Min(|T^p|, |T^q|)$. Obviously, the runtime order of Step 3 and Step 4 are $O(|T^p| + |T^q|)$ and $O(|T^p| \times |T^q|)$, respectively. Finally, the total runtime of the proposed algorithm is within the order of $O(|T^p|, |T^q| \times Min(|T^p|, |T^q|))$ since we need to keep the term with highest order and we can forget about constant coefficiants. The calculated runtime complexity is also investigated in the empirical runtime analysis section.

5 Evaluation Frameworks Design

In this section, we design clustering and classification frameworks to evaluate the proposed distance function (EST) performance against other tree distance functions. To implement these frameworks, k-medoid [12] is used for the clustering; and KNN [13] and SVM [14] are utilized for the classification framework.

5.1 Data Sets

Four different labeled data sets (three real world data sets and one synthetic data set) are utilized in this research to investigate the performance of the distance functions to prevent biased results.

The first real data set is CSLOG, available at [32]; it has appeared in a number of publications including [1], [3] and [16]. Each tree in this data set represents the behavior of a user visiting a website, where each node in the tree indicates a webpage of the website. Each tree is labeled either "edu" (visitor is from edu domain) or "other". Further, this data set contains three weeks of information separated as three data sets, presented in Table 1.

The second real data set is the ACM SIGMOD records [33] from March 1999. This data set is also utilized in several works such as [4] and [15]. Each tree in this data set is presented as an XML file. We removed one of XML files, named "a.xml", since its name was not in accordance with other

XML files and it was always miss-classified/miss-clustered. Therefore, the data set size is reduced to 987 trees.

The third real data set is called Treebank. The original data set is one huge tree, English sentences from the Wall Street Journal, tagged with parts of speech [34] like "S" (sentence), "NP" (noun phrase), "VP" (verb phrase), "PP" (prepositional phrase), "ADJP" (adjective phrase), and "ADVP" (adverb phrase). To produce a useful labeled data set for clustering and classification, we separated subtrees as samples of the classes. We considered three cases as indicated in Table 1: 1) The "NP" and "VP" subtrees as a two class data set; 2) five English phrases ("NP", "VP", "PP", "ADJP", and "ADVP") as a five class data set; and 3) five English phrases plus sentences as a six class data set. These modified data sets are available at [35]. The Treebank data sets are the largest data sets among the real data sets in this paper. Since their sizes are very large, to have a feasible runtime, we performed random sampling where 1000 random trees are selected for two class data set. In the case of the five and six class data sets, we selected 100 random trees for each class. To prevent any biased results, the whole process of random sampling and performing the experiments is repeated 100 times and the average of trials are reported. Further, a statistical analysis is performed in Section 6.4 which provides evidences on validity of these statistical experiments.

These real data sets have appeared in a number of publications [1], [3], [4], [15], [16] none of which question the veracity of these data sets. Again, we find no real evidence of miss-labeling, in these data sets and hence we believe that they are highly-accurate sources of information.

To generate the synthetic data sets, we considered data sets with different numbers of classes (2, 3, 5, 8, and 20 classes), so we can study the performance of the distance functions with respect to the number of classes in the data set. Accordingly, we generated five synthetic data sets randomly for classification and another five data sets for clustering [35]. Each synthetic data set for clustering has 50 trees for each class. The classification of the synthetic data sets has two sub-components, one for training and one for testing; each contains 50 trees for each class. Generating these data sets is a two-step process:

Step 1: Assuming we are generating a data set with N_c classes, we generate N_c labeled mother trees, namely T^{m_i} where $1 \leq i \leq N_c$. Each T^{m_i} is generated randomly where to add a new node to the tree, one of the pre-generated nodes is selected randomly and the new node is added to the selected node as a child until $|T^{m_i}| = 50$ is reached. After generating a tree with 50 nodes randomly, each node is randomly labeled from a pool of 30 possible node labels. The size of 50 for mother trees was selected since significantly larger tree sizes was not feasible with respect to runtime of the classification and clustering trials and are believed to be a reasonable representation of many situations found in computer applications. Please note that in the case of the 20 class data set, we reduced the initial size of trees to 10 to have a manageable runtime.

Step 2: After generating the mother trees, a synthetic data set is generated by producing trees using the mother trees as follows: To generate trees in accordance with the ith mother tree, T^{m_i}, we go through all the nodes in T^{m_i} and each node, namely t_j, is edited with the probability ofρ. For each t_j, the edit operation is randomly selected with equal probability from one of five edit operations. Let T^p be a subtree of T^{m_i} rooted at t_j and includes all of the children of t_j; also let T^q be a subtree of T^{m_i} rooted at a random node which includes all of the children of that random node. Further, T^r is selected randomly like T^q, but from T^{m_k} rather than T^{m_i} where T^{m_k} is a randomly chosen mother tree other than T^{m_i}. The five edit operation are: 1)T^p is removed; 2) T^p is replaced with T^q; 3) T^p is replaced with T^r; 4) T^q is added to the root of T^p as a child; and 5) T^r is added to the root of T^p as a child. In fact, a tree in the ith class is a combination of ith mother tree and other mother trees. For the classification trials, $\rho = 0.5$ was selected; and for the clustering trials, it is selected as 0.25. ρ is chosen lower for the clustering trials, since the clustering results are more sensitive than the classification results; so to keep the results of clustering around 80 percent; accurate, ρ is reduced. Finally, since the synthetic data set generation is a probabilistic process, to prevent any biased results, the whole process of generating data sets and conducting the trials is repeated 100 times and the average of trials are reported as the synthetic data sets results.

5.2 Clustering Framework

The k-medoid [12] clustering technique is used as one of the evaluation approaches. The reason why k-medoid is chosen over other clustering techniques, like the classic k-means, is the limitation enforced by our tree data type. Unlike usual machine learning problems, we cannot simply model trees as a set of features and consequently we cannot define operations such as averaging on trees which is required in most of the clustering techniques such as k-means which partitions the data into Voronoi regions [36]. In our case, the only operation defined on trees is the distance between two trees acquired using a distance function. K-medoid is similar to k-means, however, a tree is selected as the center of the cluster, called a medoid, rather than the average of the trees. Partitioning Around Medoids (PAM) algorithm [12] is utilized for clustering. Further, to determine the initial medoids, we selected the first medoid randomly and then trees with highest distance from previously selected medoids. Since the k-medoid approach may find a local optimum rather than the global optimum, the clustering process is repeated 10 times with different initial medoids and the results with minimum cost are selected as the final results. After the clustering is completed, we need to assign a label to each cluster as a predicted label for evaluation purposes. A predicted label is determined as the real label that has the greatest population within that cluster.

5.3 Classification Framework

Weighted KNN [13] and kernel based SVM [14] are utilized to perform the evaluation of the distance functions with respect to the classification applications. Similar to clustering discussion, not all the classification techniques work on trees, since we only have a distance between trees, not their features. As a result, only methods like

KNN that purely work based on distance functions; and kernel based classification approaches such as SVM, which map the input space into higher dimensions, can be utilized for tree classification problems.

KNN is a classic classification technique where the K nearest neighbors to the test data are identified from the set of training data, and then, in a voting process with S^* as a weight, the predicted class for the test data is determined. K is chosen as nine in all the trial runs since we observed it generates the best results.

Kernel based SVM [14] is a state-of-the-art classification technique which maps the input space into higher dimensions and generates support vectors in the new space. The mapping is performed based on the kernel function. Beside SVM's kernel function which is set to S^* in all the evaluations, we need to specify the penalty factor (C) and epsilon (ε). C controls the over/under fitting in the training stage [14] and is set to 2 for all the real data sets, and 4 for all the synthetic data sets in this research. Further, ε which has an effect on the smoothness of the SVM's response and the number of support vectors, is selected as 0.001 in all the experiments. Since SVM originally works only on two class problems, we utilized the one-versus-all technique [37] to extend the SVM classification to multi-classes.

Unlike clustering, we need to train these classifiers, and then perform a classification on another date set. In the case of the CSLOG data sets, we performed three experiments, distinguished as CSLOG_2C_WEEK12, 23, and 31 where the last two digits refer to the training and the test data, respectively. In contrast, in the case of the SIGMOD and Treebank data sets, we utilized 10-fold cross validation approach.

5.4 Clustering and Classification Evaluation

After the classification experiments, we have predicted and real labels for each test tree. Similarly, as discussed in the clustering section, after a clustering experiment, each cluster is assigned a label. Hence, each tree has a predicted label beside its real label. Now that we have predicted and real labels of each tree, the evaluation is performed in terms of 1) accuracy; 2) Weighted Average of F-measures (WAF); and 3) runtime.

Accuracy is simply defined as the number of correctly clustered/classified trees over the total number of trees. F-measure is a popular information retrieval metric that is defined for each class and integrates recall and precision using the harmonic mean. Let C_i be the set of all the trees in class i; and let P_i be the set of all the trees predicted (clustered or classified) to be in class i, where $1 \leq i \leq N_c$. The recall, precision, and f-measure are defined as:

$$Recall_i = \frac{a_i}{|C_i|}, Precision_i = \frac{b_i}{|P_i|}, \\ F\ Measure_i = \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i}, \tag{9}$$

where a_i and b_i denote the number of correctly clustered/classified trees in C_i and P_i, respectively. Since the number of classes is different with respect to different

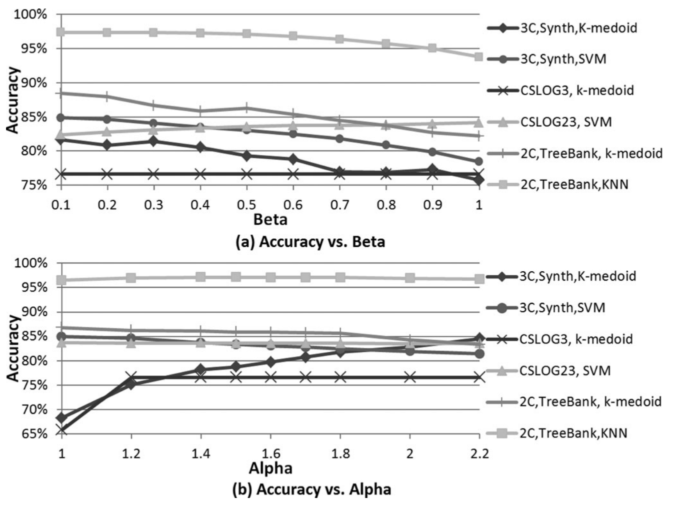


Fig. 8. The accuracy of EST similarity function against α and β.

data sets and space is limited, providing a f-measure for all classes separately is not feasible. Therefore, the WAF, as used in WEKA [38], is presented regarding clustering/classification evaluation, where it is defined as:

$$WAF = \sum_{i=1}^{N_c} \frac{|C_i|}{number\ of\ all\ trees} \times FMeasure_i, \tag{10}$$

where N_c denotes the number of classes in a data set. The experiments within this study were conducted using Java 7 (64bit). The hardware platform, where the experiments have been executed, was an Intel dual-core Processor E6300 (2.8 GHz) with 8 GB of RAM. The k-medoid and KNN are bespoke implementations; and the libsvm Java SVM library [39] was adapted for the tree classification applications.

5.5 Distance Function's Parameters

The proposed EST approach includes two parameters, α and β, that need to be adjusted. Obviously, they can be adjusted for every single experiment to achieve the optimum performance. However, for all the experiments, we have fixed the values of the parameters to produce an equivalence with the other distance functions which are (relatively) parameter free.

As discussed in Section 4.2, β reflects the relative position of the mapped subtrees and it can be adjusted between 0 and 1. However, we have no any formal mechanism for estimating β at the moment. Hence, we set β to the neutral value of 0.5 which seems a good balancing point according to the sensitivity analysis presented in Fig. 8a. Fig. 8a demonstrates accuracy versus β, we did not include all the experiments in this figure to prevent a busy figure. For each data set, one clustering and one classification experiment is selected. In addition, we did not include SIGMOD data set as it produces 100 percent accuracy for all values of β. One can observe from Fig. 8a that with the increasing β, accuracy is reducing for some of the experiments and increasing for some other. Therefore, the neutral value of 0.5 is selected for all experiments. Finally, WAF sensitivity analysis is not presented as it was similar to accuracy results.

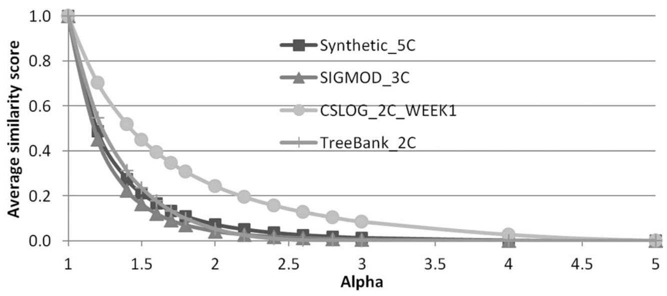


Fig. 9. The average similarity of EST similarity function against α.

α reflects the importance of the size of the mapped subtrees. Our formulation can be thought of as a variation on the well-known Minkowski distance [31]. As explained in [31], the optimal value of α will vary with the domain of application and hence no universal approach for optimally estimating α exists. In the absence of problem-specific knowledge, it is believed that α can be estimated in our formulation by considering the variation of the average similarity against α ($\alpha \geq 1$). As $\alpha \to 1$, the algorithm overly weights the impact of small trees compared to the impact of large trees. These small trees are minor in terms of the "big picture"; however, their existence or not, can have a large impact on the similarity result and hence we can view the metric as becoming "numerical unstable" as $\alpha \to 1$. Conversely, as $\alpha \to \infty$, the distance metric loses discrimination power. Large trees dominate and substantial variations on small tress have little or no impact on the resultant similarity score. Hence, the selection of α is equivalent to finding the balancing point which minimizes these two undesirable behaviors.

Let α_i represent the ideal balancing point. If we consider a plot of the average similarity against α as shown in Fig. 9, we can define the plot in terms of: C1—the curve between $\alpha = 1$ and $\alpha = \alpha_i$; and C2—the curve between $\alpha = \alpha_i$ and $\alpha = \infty$. From above, C1 can be characterized as a curve where the average similarity changes significant with small changes in α; and C2 as a curve where the similarity changes slowly (in fact, we believe that C2 can be modeled as a linear segment implying no curvature exists across C2). This model is again well-known and is perhaps most commonly used in the Scree test [40].

Accordingly we need to estimate the balancing point (elbow) in Fig. 9 (Only a subset of the data is shown in the figure to increase clarity). In addition, we have scaled the curves between 0 and 1. Several approaches exist to approximate the elbow point, e.g., the Angle-based technique [41], the Menger Curvature method [42] which is good for continuous data, and the Kneedle technique [43] which works for both discrete and continues data. None of these approaches are perfect in the presence of noise. Satopaa et al. [43] discuss that the Menger Curvature method is sensitive to noise, while the Kneedle technique produces better results. On average, all the methods give very similar results. The Kneedle technique [43] is believed to be more robust to noise and works for discrete data. Therefore, we utilize the Kneedle technique [43] to estimate the elbow point. The elbow point for each curve was slightly different with the average value of $\alpha = 1.6$. We utilized this value for all the experiments in this research and recommend this value in situations where limited data exists stopping the re-estimation of α.

TABLE 2
The Clustering Results for all Case Studies in the Terms of Accuracy, Weighted Average of F-Measure (WAF), and Runtime

	Data set	EST	TED	Entropy	Path	Multiset	IST
Accuracy, %	CSLOG_2C_WEEK1	**73.6**	60.0	61.9	**73.6**	63.7	62.6
	CSLOG_2C_WEEK2	**74.8**	60.0	63.0	64.1	62.2	65.9
	CSLOG_2C_WEEK3	**76.6**	63.0	62.5	69.8	63.4	62.1
	SIGMOD_3C	**100**	99.9	54.1	99.9	99.9	**100**
	TREEBANK_2C	**85.9**	76.1	74.7	75.4	77.0	80.6
	TREEBANK_5C	**70.3**	58.2	54.6	61.3	57.1	62.8
	TREEBANK_6C	**64.2**	54.8	51.0	54.4	48.5	54.7
	SYN_2C_CLUSTER	**87.4**	61.8	81.7	78.0	63.9	82.1
	SYN_3C_CLUSTER	79.7	60.6	72.5	68.9	56.8	**81.2**
	SYN_5C_CLUSTER	**78.6**	62.7	69.1	62.9	47.3	75.3
	SYN_8C_CLUSTER	**79.1**	64.1	69.6	60.5	42.0	75.2
	SYN_20C_CLUSTE	**77.3**	62.8	72.4	58.0	37.0	71.5
WAF, %	CSLOG_2C_WEEK1	**69.4**	62.5	59.1	**69.4**	60.1	59.4
	CSLOG_2C_WEEK2	**71.6**	63.0	60.9	66.1	60.4	62.3
	CSLOG_2C_WEEK3	**72.2**	65.3	59.9	63.7	60.4	59.7
	SIGMOD_3C	**100**	99.9	68.6	99.9	99.9	**100**
	TREEBANK_2C	**85.8**	76.1	74.0	74.2	76.2	80.3
	TREEBANK_5C	**70.9**	58.7	55.3	61.8	57.5	63.3
	TREEBANK_6C	**63.7**	55.5	50.7	53.6	47.7	54.0
	SYN_2C_CLUSTER	**86.8**	56.7	81.0	76.4	61.1	81.7
	SYN_3C_CLUSTER	76.4	55.5	68.4	63.6	51.0	**80.6**
	SYN_5C_CLUSTER	**75.1**	59.5	64.2	56.3	40.1	73.5
	SYN_8C_CLUSTER	**76.4**	61.7	65.1	54.0	34.3	73.9
	SYN_20C_CLUSTE	**76.1**	62.2	71.1	56.8	35.7	71.1
Runtime, minutes	CSLOG_2C_WEEK1	**4.8**	13.5	110.7	102.8	23.1	66.0
	CSLOG_2C_WEEK2	**6.5**	17.7	146.4	138.5	29.8	82.4
	CSLOG_2C_WEEK3	**6.3**	15.1	86.4	76.2	26.8	70.2
	SIGMOD_3C	2.7	11.6	2.1	2.0	**1.7**	156.2
	TREEBANK_2C	**32.1**	144.7	85.7	70.5	35.5	244.8
	TREEBANK_5C	**5.9**	17.9	15.5	12.2	6.7	32.9
	TREEBANK_6C	**15.9**	55.6	39.8	32.5	19.4	98.4
	SYN_2C_CLUSTER	**4.2**	26.3	7.3	6.0	4.8	67.2
	SYN_3C_CLUSTER	**9.5**	58.0	15.4	12.6	11.4	141.7
	SYN_5C_CLUSTER	**23.9**	156.0	42.6	35.1	33.6	511.8
	SYN_8C_CLUSTER	**59.5**	400.8	109.1	89.7	89.8	1361.3
	SYN_20C_CLUSTE	**66.6**	87.3	80.6	79.1	70.7	165.5

Further, we performed a sensitivity analysis of accuracy against α as presented in Fig. 8b. Similar to β, we select a few experiments to prevent a busy figure. According to this figure, the calculated value of 1.6 for α seems to be a balancing point for all the experiments.

Finally, the costs of edit operations (γ) are considered as the unit scalar regarding TED and IST for all experiments. The Entropy, Path, and Multiset distance functions have no parameters to discuss.

6 Experimental Results and Discussion

6.1 K-medoid Clustering Results

Table 2 represents the k-medoid clustering results with respect to all data sets discussed in Section 5.1. The purpose of this table is to compare the performance of the proposed EST approach against the previous distance functions when they are used as the core in clustering applications. This table has three parts associated to accuracy, WAF, and runtime evaluations. Further, the result of the best evaluated distance function is bolded.

As indicated in Table 2, the proposed EST has outperformed other distance functions in most of the investigated situations in terms of accuracy, WAF, and runtime. In the case of the three CSLOG data sets, EST has significantly improved results, over 10 percent in accuracy, except for the

TABLE 3
The KNN Classification Results for all Case Studies in the Terms of Accuracy, Weighted Average of F-Measure, and Runtime

	Data set	EST	TED	Entropy	Path	Multiset	IST
Accuracy, %	CSLOG_2C_WEEK12	**83.4**	83.1	83.3	83.2	83.0	83.2
	CSLOG_2C_WEEK23	**84.1**	83.1	83.9	83.9	83.7	83.8
	CSLOG_2C_WEEK31	**83.2**	82.4	83.0	83.0	83.0	83.0
	SIGMOD_3C	**100**	**100**	**100**	**100**	99.9	**100**
	TREEBANK_2C	**97.0**	96.9	93.9	94.9	90.1	95.5
	TREEBANK_5C	**88.8**	86.3	79.2	81.9	69.6	83.0
	TREEBANK_6C	**87.2**	84.8	75.5	78.2	65.7	80.4
	SYN_2C_CLASIFY	**82.9**	78.1	77.6	76.9	72.4	80.5
	SYN_3C_CLASIFY	**75.1**	67.4	70.7	70.0	60.6	71.4
	SYN_5C_CLASIFY	**70.3**	61.6	67.8	66.4	55.6	66.4
	SYN_8C_CLASIFY	**68.7**	59.1	65.8	65.1	53.0	63.9
	SYN_20C_CLASIFY	**68.9**	60.5	62.0	59.3	37.9	63.1
WAF, %	CSLOG_2C_WEEK12	**82.4**	81.8	82.3	82.2	81.9	82.2
	CSLOG_2C_WEEK23	**83.2**	81.7	82.9	83.0	82.6	82.8
	CSLOG_2C_WEEK31	**82.0**	80.9	81.9	81.9	81.8	81.9
	SIGMOD_3C	**100**	**100**	**100**	**100**	99.9	**100**
	TREEBANK_2C	**97.0**	96.9	93.9	94.9	90.1	95.5
	TREEBANK_5C	**89.0**	86.5	79.6	82.3	70.0	83.2
	TREEBANK_6C	**87.3**	84.9	75.4	78.4	65.5	80.3
	SYN_2C_CLASIFY	**82.8**	77.9	77.3	76.7	72.2	80.4
	SYN_3C_CLASIFY	**75.0**	67.4	70.5	69.9	60.4	71.4
	SYN_5C_CLASIFY	**70.3**	61.7	67.7	66.3	55.5	66.4
	SYN_8C_CLASIFY	**68.8**	59.3	65.9	65.2	52.9	64.0
	SYN_20C_CLASIFY	**69.4**	60.9	62.3	59.6	37.8	63.6
Runtime, minutes	CSLOG_2C_WEEK12	**5.8**	15.3	126.7	117.5	26.3	75.5
	CSLOG_2C_WEEK23	**5.6**	14.3	119.9	113.5	24.3	67.4
	CSLOG_2C_WEEK31	**6.7**	15.8	91.5	80.7	28.2	74.3
	SIGMOD_3C	2.6	11.4	2.0	1.8	**1.6**	156.0
	TREEBANK_2C	**29.8**	142.3	83.2	68.0	33.2	242.5
	TREEBANK_5C	**3.5**	15.4	13.1	10.0	4.5	30.6
	TREEBANK_6C	**8.4**	47.2	31.5	24.7	11.8	90.4
	SYN_2C_CLASIFY	**4.0**	28.2	8.0	6.7	5.3	73.7
	SYN_3C_CLASIFY	**9.3**	64.0	18.3	15.1	13.2	155.9
	SYN_5C_CLASIFY	**22.2**	173.0	47.2	38.8	34.8	420.9
	SYN_8C_CLASIFY	**58.9**	429.8	116.5	94.9	90.7	1055.4
	SYN_20C_CLASIFY	**26.8**	55.3	48.2	36.4	28.9	185.7

TABLE 4
The SVM Classification Results for all Case Studies in the Terms of Accuracy, Weighted Average of F-Measure, and Runtime

	Data set	EST	TED	Entropy	Path	Multiset	IST
Accuracy, %	CSLOG_2C_WEEK12	**83.5**	70.8	83.2	82.9	83.0	82.4
	CSLOG_2C_WEEK23	**83.6**	68.1	83.5	83.1	83.0	82.9
	CSLOG_2C_WEEK31	**82.8**	68.3	82.5	82.3	82.2	81.4
	SIGMOD_3C	**100**	**100**	**100**	**100**	**100**	**100**
	TREEBANK_2C	**99.9**	**99.9**	99.8	99.8	98.6	**99.9**
	TREEBANK_5C	**99.7**	**99.7**	99.3	99.2	97.4	99.5
	TREEBANK_6C	**99.7**	**99.7**	99.0	99.0	96.8	99.2
	SYN_2C_CLASIFY	88.0	87.0	86.2	84.8	84.9	**88.3**
	SYN_3C_CLASIFY	**83.1**	82.2	79.7	79.0	76.9	82.6
	SYN_5C_CLASIFY	**79.8**	78.9	77.1	75.6	72.3	79.7
	SYN_8C_CLASIFY	**78.6**	77.4	75.6	74.2	69.4	77.8
	SYN_20C_CLASIFY	**71.6**	64.0	66.5	65.3	50.3	59.0
WAF, %	CSLOG_2C_WEEK12	**82.3**	70.5	82.0	81.4	81.6	81.2
	CSLOG_2C_WEEK23	**82.3**	68.0	**82.3**	81.7	81.5	81.7
	CSLOG_2C_WEEK31	**81.4**	67.8	81.1	80.6	80.6	80.0
	SIGMOD_3C	**100**	**100**	**100**	**100**	**100**	**100**
	TREEBANK_2C	**99.9**	**99.9**	99.8	99.8	98.6	**99.9**
	TREEBANK_5C	**99.7**	**99.7**	99.3	99.2	97.4	99.5
	TREEBANK_6C	**99.7**	**99.7**	99.0	99.0	96.8	99.2
	SYN_2C_CLASIFY	87.9	87.0	86.2	84.8	84.9	**88.3**
	SYN_3C_CLASIFY	**83.0**	82.2	79.6	79.0	76.9	82.6
	SYN_5C_CLASIFY	**79.8**	78.9	77.1	75.6	72.3	79.7
	SYN_8C_CLASIFY	**78.6**	77.5	75.6	74.2	69.4	77.8
	SYN_20C_CLASIFY	**72.0**	64.3	66.8	65.7	50.3	59.3
Runtime, minutes	CSLOG_2C_WEEK12	**11.1**	28.7	237.4	220.2	49.5	141.6
	CSLOG_2C_WEEK23	**12.6**	31.9	266.2	252.1	54.1	149.6
	CSLOG_2C_WEEK31	**13.1**	31.1	177.8	157.0	54.9	144.4
	SIGMOD_3C	2.8	11.4	2.1	2.0	**1.6**	156.4
	TREEBANK_2C	**30.1**	142.9	83.5	68.4	33.4	243.1
	TREEBANK_5C	**4.2**	16.5	13.9	10.8	5.2	31.5
	TREEBANK_6C	**9.7**	49.0	32.8	26.2	12.8	91.9
	SYN_2C_CLASIFY	**8.9**	58.8	16.4	13.6	10.9	148.2
	SYN_3C_CLASIFY	**20.1**	127.4	35.2	29.2	26.0	325.7
	SYN_5C_CLASIFY	**53.2**	351.8	94.9	79.2	65.3	895.7
	SYN_8C_CLASIFY	**123.1**	875.2	233.8	194.8	174.7	2249.1
	SYN_20C_CLASIFY	**54.5**	112.2	97.3	73.4	58.5	376.7

Path distance with respect to CSLOG_2C_WEEK1. With respect to SIGMOD_3C data set, EST and IST produce the perfect result; TED, Path, and Multiset have only one miss-clustered tree; finally, the Entropy approach produces a very poor performance. EST also significantly outperformed other approaches (over 5 percent) with respect to the Treebank data sets. Regarding the synthetic data sets, the Multiset approach has the worst result and its accuracy and WAF significantly reduces as the number of classes grows. EST produced the best results in most cases with IST in second place. Apart from TED, the performances of all the distance functions are degraded as the number of classes increases. However, this result is not an advantage for TED since its uniformly poor performance. Finally, the runtime results indicate that EST has the best efficiency in term of runtime, except for the SIGMOD data set. This result makes the proposed approach the best for real time applications. In contrast, IST and Entropy produce the largest execution times.

6.2 KNN Classification Results

The KNN classification results with respect to all data sets are represented in Table 3 where the performances of the distance functions are investigated. The CSLOG data sets' results suggest that all the distance functions have a similar performance with regard to accuracy and WAF measures. Apart from the Multiset approach, all other distance functions produced a perfect classification regarding the SIGMOD data set. The Multiset approach produced one miss-classification. EST and then TED produced the best results with respect to the Treebank data sets. In case of the synthetic data sets, the EST approach produces the most accurate results; and the largest WAF measures. IST has the second most impressive results in terms of accuracy and WAF. Similar to the clustering runtime results, EST has the lowest runtime complexity for all data sets except SIGMOD where the Multiset approach produces the lowest runtime. Again, similar to the clustering runtime results, Entropy and IST have the largest runtime performances.

6.3 SVM Classification Results

The SVM classification results are presented in Table 4; they are similar to the KNN results. The differences include the now perfect result for the Multiset distance function in case of the SIGMOD data set. In case of the Treebank data sets, all the approaches produce very good results. In addition, IST produced the highest accuracy with respect to SYN_2C_CLASIFY data set. However, the EST has still the greatest accuracy with regard to the other synthetic data sets. The runtime results with respect to lowest and largest runtime are similar to the KNN classification and the clustering case studies. These suggest the proposed EST has, in general, the lowest runtime complexity. Although the comparison between KNN and SVM is not within the scope of this research, one can observe that SVM possesses a better accuracy and WAF on average.

TABLE 5
Cohen's Effect Size as Well as Corresponding Values for Percentile Standing and Percent of Non-Overlapped Portion of Two Populations

Cohen's Description	Effect Size	Percentile Standing	Percent of Non-overlap
	2.0	97.7	81.1%
	1.5	93.3	70.7%
	1.0	84	55.4%
Large	0.8	79	47.4%
Medium	0.5	69	33.0%
Small	0.2	58	14.7%
	0.0	50	0.0%

6.4 Statistical Analysis of Results

As explained in Section 5.1, the results of the Treebank and Synthetic data sets are averaged over 100 trial runs. Therefore, we have a population of 100 results for each experiment which allows us to perform a test of statistical significance (z-test, one-tailed, our working hypothesis is that the EST will produce superior results) with a conservative type I error of 0.01. Further, we have calculated effect size (Cohen's method [36]) which estimates the "size" discrepancy between two statistical populations. Cohen defines the standard value of an effect size as small (0.2), medium (0.5), and large (0.8). Effect size can also be interpreted as the average percentile standing which indicates the relative position of the two populations. Similarly, effect sizes can be interpreted in terms of the percent of the non-overlapped portion of the populations. Corresponding values are presented in Table 5.

Accordingly, Table 6 represents the effect size for accuracy of EST against all the previous approaches. In this table, a positive value of effect size indicates that EST outperformed that method. The "*" beside an effect size indicates the result of the z-test where a significant difference exist at the 0.01 level. The results indicate that in most of the experiments EST statistically significant outperforms other approaches.

6.5 Empirical Runtime Analysis

In addition to accuracy and WAF, the computational cost of an algorithm is an important factor in practical applications. The runtime of the clustering and classification experiments are reported in Tables 2, 3, and 4. To further empirically compare the distance functions' runtime, we measure the distance calculation runtime with respect to different tree sizes. Tree sizes between 5 and 100 with step size of 5 have been investigated where both trees are generated randomly as described in the synthetic tree generation section. In addition, the hardware platform is in accordance with the platform described at the end of Section 5.4; and again, Java 7 (64 bit) is utilized to implement the source code. The runtime measurement is performed 1,000 times and the average distance function execution times are presented in Fig. 10 in milliseconds.

The IST distance function produced the largest runtime followed by TED and then the Multiset approach. The EST, Path, and Entropy have the best runtime; all three approaches produce broadly similar results.

TABLE 6
The Effect Size between Accuracy of the EST and Previous Approaches

	Data set	TED	Entropy	Path	Multiset	IST
K-medoid	TREEBANK_2C	5.81*	2.10*	1.17*	1.60*	1.65*
	TREEBANK_5C	2.59*	3.83*	2.68*	3.91*	2.22*
	TREEBANK_6C	2.46*	3.38*	2.72*	4.81*	2.62*
	SYN_2C_CLUSTER	2.40*	0.79*	0.98*	3.69*	0.78*
	SYN_3C_CLUSTER	1.69*	0.38*	0.69*	2.69*	-0.11
	SYN_5C_CLUSTER	1.95*	0.77*	1.47*	5.17*	0.28*
	SYN_8C_CLUSTER	2.02*	0.99*	1.97*	6.95*	0.42*
	SYN_20C_CLUSTER	3.33*	1.10*	4.25*	9.86*	1.29*
KNN	TREEBANK_2C	0.17	4.84*	3.53*	8.79*	2.46*
	TREEBANK_5C	1.70*	5.79*	4.17*	9.76*	3.74*
	TREEBANK_6C	1.66*	6.96*	5.83*	12.30*	4.38*
	SYN_2C_CLASIFY	1.08*	1.03*	1.38*	2.27*	0.50*
	SYN_3C_CLASIFY	1.84*	0.96*	1.26*	3.40*	0.87*
	SYN_5C_CLASIFY	2.20*	0.67*	1.06*	3.91*	1.03*
	SYN_8C_CLASIFY	3.57*	1.01*	1.31*	5.39*	1.73*
	SYN_20C_CLASIFY	4.25*	3.54*	5.02*	14.90*	2.91*
SVM	TREEBANK_2C	0.00	0.82*	0.78*	1.07*	0.00
	TREEBANK_5C	0.00	1.30*	1.46*	4.24*	0.76*
	TREEBANK_6C	0.00	2.09*	2.24*	5.97*	1.55*
	SYN_2C_CLASIFY	0.24*	0.49*	0.83*	0.91*	-0.09
	SYN_3C_CLASIFY	0.28*	1.04*	1.25*	2.03*	0.14
	SYN_5C_CLASIFY	0.29*	0.89*	1.36*	2.72*	0.02
	SYN_8C_CLASIFY	0.58*	1.47*	2.06*	5.00*	0.40*
	SYN_20C_CLASIFY	3.54*	2.66*	3.23*	9.91*	5.70*

* *Indicates the result of the z-test where a significant difference exist at the 0.01 level.*

7 Conclusion and Discussion

In this research, the novel EST similarity function has been proposed for the domain of tree structured data comparison with the aim of increasing the effectiveness of applications utilizing tree distance or similarity functions. This new approach seeks to resolve the problems and limitations of previous approaches, as discussed in Section 4.1. In addition, the new approach must enhance applications where a tree distance function is utilized. To achieve this goal, we first extensively analyze other distance functions. Then, we identified situations where the studied distance functions have poor performance; and finally we propose the EST approach. The proposed EST approach preserves the structure of the trees by mapping subtrees rather than nodes. EST generalizes the edit base distances and mappings provided in Section 3.1 by breaking the one-to-one and order preserving mapping rules. Further, it introduces new rules for subtree mapping provided in Section 4.2.

An extensive experimental study has been performed to evaluate the performance of the proposed similarity function against previous research. Clustering and classification

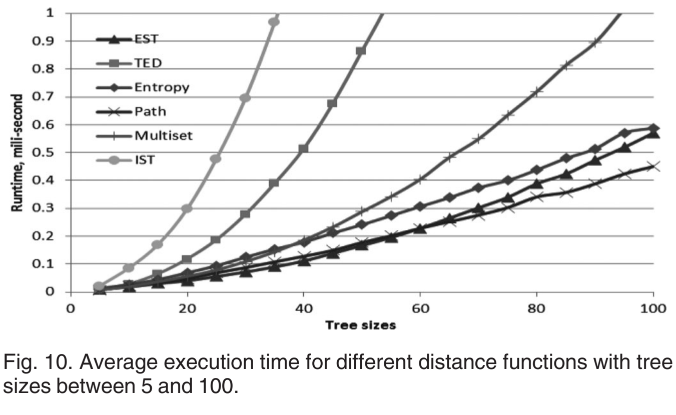


Fig. 10. Average execution time for different distance functions with tree sizes between 5 and 100.

frameworks are designed to perform an unbiased evaluation according to K-medoid, KNN, and SVM along with four distinct data sets. The real-world data sets have appeared in a number of publications [1], [3], [4], [15], [16] and hence they are deemed to be reliable source of information. Further, using synthetic data sets, we investigated the effect of varying the number of classes in the evaluation. This extensive evaluation framework is one of the advantages of this research over previous researches such as [1], [3], [4], [15], and [16].

The results of the experimental studies demonstrate that the EST approach is superior to the other approaches with respect to classification and clustering applications. To evaluate the performance, accuracy and WAF, are used in Tables 2, 3, and 4, where, in general, EST is demonstrated to be a better option for the clustering and classification of tree structured data. However, the performance of a distance function varies with the domain of application; and hence, we cannot generalize the superior performance of EST to all domains of application.

The computational cost of a tree distance function should be carefully considered for practical applications. Given T^p and T^q as the input trees to the distance function, we calculated the runtime order of the EST as $O(|T^p| \times |T^q| \times Min(|T^p|, |T^q|))$. Further, the runtime of all the clustering and classification experiments are measured where the proposed EST outperformed all other distance functions with respect to all data sets except SIGMOD. In addition, an empirical analysis has been performed to compare the runtime of EST versus other distance functions in different tree sizes. The result of this empirical investigation suggests that the runtime efficiency of EST, Entropy, and Path are better than the other distance functions. Accordingly, the conclusion can be drawn that the proposed EST is an appropriate approach for computationally restricted and real time applications.

The Java source code of the proposed EST is available at [35] to allow for the repeatability of experiments and extendibility of this research. In addition, the implemented source code of the TED, Path, Entropy, Multiset, and IST along with the source code developed for tree based k-medoid, KNN, and SVM are available from the same address.

Finally, although further studies are required to validate the use of the EST similarity function in real-life applications, EST has been demonstrated to have a superior performance against TED, IST, Path, Entropy, and Multiset distance functions with respect to classification and clustering applications. Although the result of this research suggests an improvement in the quality of tree comparison, the tree distance function still has room for improvement considering its numerous applications discussed in the introduction. Future research in similarity/distance measurement of tree structured data should focus on lowering computational complexity, improving the quality, and investigating the distance functions on other real-world applications.

REFERENCES

[1] M.J. Zaki, "Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 8, pp. 1021-1035, Aug. 2005.

[2] J. Punin, M. Krishnamoorthy, and M. Zaki, "LOGML: Log Markup Language for Web Usage Mining," *Proc. Revised Papers from the Third Int'l Workshop Mining Web Log Data Across All Customers Touch Points (WEBKDD '01)*, pp. 273-294, 2002.

[3] M.J. Zaki and C.C. Aggarwal, "XRules: An Effective Structural Classifier for XML Data," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 316-325, 2003.

[4] W. Lian, D. W. -l. Cheung, N. Mamoulis, and S.-M. Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 1, pp. 82-96, Jan. 2004.

[5] M. Kouylekov and B. Magnini, "Recognizing Textual Entailment with Tree Edit Distance Algorithms," *Proc. First Challenge Workshop Recognising Textual Entailment*, pp. 17-20, 2005.

[6] A. Mesbah and M.R. Prasad, "Automated Cross-Browser Compatibility Testing," *Proc. 33rd Int'l Conf. Software Eng. (ICSE)*, pp. 561-570, 2011.

[7] A. Mesbah, A. van Deursen, and D. Roest, "Invariant-Based Automatic Testing of Modern Web Applications," *IEEE Trans. Software Eng.*, vol. 38, no. 1, pp. 35-53, Jan./Feb. 2012.

[8] R. Connor, F. Simeoni, M. Iakovos, and R. Moss, "A Bounded Distance Metric for Comparing Tree Structure," *Information Systems*, vol. 36, no. 4, pp. 748-764, 2011.

[9] D. Buttler, "A Short Survey of Document Structure Similarity Algorithms," *Proc. Fifth Int'l Conf. Internet Computing*, 2004.

[10] P. Bille, "A Survey on Tree Edit Distance and Related Problems," *Theoretical Computer Science*, vol. 337, no. 1–3, pp. 217-239, 2005.

[11] A. Muller-Molina, K. Hirata, and T. Shinohara, "A Tree Distance Function Based on Multi-Sets," *New Frontiers in Applied Data Mining*, S. Chawla, T. Washio, S. Minato, S. Tsumoto, T. Onoda, S. Yamada, and A. Inokuchi, eds., vol. 5433, pp. 87-98, Springer, 2009.

[12] L. Kaufman et al., *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 39, John Wiley & Sons Online Library, 1990.

[13] W. Zuo, D. Zhang, and K. Wang, "On Kernel Difference-Weighted k-Nearest Neighbor Classification," *Pattern Analysis & Applications*, vol. 11, no. 3, pp. 247-257, 2008.

[14] E. Alpaydin, "Support Vector Machines," *Introduction to Machine Learning*, second ed., pp. 218-225, MIT Press, 2004.

[15] C.C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. Zaki, "Xproj: A Framework for Projected Structural Clustering of XML Documents," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery Data Mining*, pp. 46-55, 2007.

[16] F. Hadzic and M. Hecker, "Alternative Approach to Tree-Structured Web Log Representation and Mining," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence and Intelligent Agent Technology (WI-IAT)* , vol. 1, pp. 235-242, 2011.

[17] K. Zhang and D. Shasha, "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems," *SIAM J. Computing*, vol. 18, no. 6, pp. 1245-1262, 1989.

[18] K.-C. Tai, "The Tree-to-Tree Correction Problem," *J. ACM*, vol. 26, no. 3, pp. 422-433, July 1979.

[19] J. T. L. Wang and K. Zhang, "Finding Similar Consensus between Trees: An Algorithm and a Distance Hierarchy," *Pattern Recognition*, vol. 34, no. 1, pp. 127-137, 2001.

[20] A. Nierman and H.V. Jagadish, "Evaluating Structural Similarity in XML Documents," *Proc. Fifth Int'l Workshop Web and Databases (WebDB '02)*, pp. 61-66, 2002.

[21] E. Tanaka and K. Tanaka, "The Tree-to-Tree Editing Problem," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 2, no. 2, pp. 221-240, 1988.

[22] G. Valiente, "An Efficient Bottom-Up Distance between Trees," *Proc. Eighth Int'l Symp. String Processing and Information Retrieval (SPIRE '01)*, pp. 212-219, 2001.

[23] K. Zhang, "Algorithms for the Constrained Editing Distance between Ordered Labeled Trees and Related Problems," *Pattern Recognition*, vol. 28, no. 3, pp. 463-474, 1995.

[24] T. Jiang, L. Wang, and K. Zhang, "Alignment of Trees—An Alternative to Tree Edit," *Theoretical Computer Science*, vol. 143, no. 1, pp. 137-148, 1995.

[25] S.M. Selkow, "The Tree-to-Tree Editing Problem," *Information Processing Letters*, vol. 6, no. 6, pp. 184-186, 1977.

[26] S.Y. Lu, "A Tree-Matching Algorithm Based on Node Splitting and Merging," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 2, pp. 249-256, Mar. 1984.

[27] S. Helmer, "Measuring the Structural Similarity of Semistructured Documents Using Entropy," *Proc. 33rd Int'l Conf. Very Large Data Bases*, pp. 1022-1032, 2007.

[28] M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitanyi, "The Similarity Metric," *IEEE Trans. Information Theory*, vol. 50, no. 12, pp. 3250-3264, Dec. 2004.

[29] R. Yang, P. Kalnis, and A.K.H. Tung, "Similarity Evaluation on Tree-Structured Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 754-765, 2005.

[30] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, "Fast Detection of XML Structural Similarity," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 2, pp. 160-175, Feb. 2005.

[31] P.J.F. Groenen and K. Jajuga, "Fuzzy Clustering with Squared Minkowski Distances," *Fuzzy Sets Systems*, vol. 120, no. 2, pp. 227-237, 2001.

[32] M.J. Zaki, "CSLOG Data Set," http://www.cs.rpi.edu/~zaki/software/logml/, 2014.

[33] "XML SIGMOD Record," http://www.sigmod.org/publications/sigmod-record/xml-edition, 2014.

[34] "Treebank Data Set," http://www.cs.washington.edu/research/xmldatasets/, 2014.

[35] A. Shahbazi and J. Miller, "Source Code and Data Sets of This Research," http://www.steam.ualberta.ca/main/Papers/ESubtree/, 2014.

[36] A. Shahbazi, A. Tappenden, and J. Miller, "Centroidal Voronoi Tessellations—A New Approach to Random Testing," *IEEE Trans. Software Eng.*, vol. 39, no. 2, pp. 163-183, Feb. 2012.

[37] R. Rifkin and A. Klautau, "In Defense of One-vs-All Classification," *J. Machine Learning Research*, vol. 5, pp. 101-141, Dec. 2004.

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The WEKA Data Mining Software: An Update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10-18, Nov. 2009.

[39] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1-27:27, 2011.

[40] R.B. Cattell, "The Scree Test for the Number of Factors," *Multivariate Behavioral Research*, vol. 1, no. 2, pp. 245-276, 1966.

[41] Q. Zhao, V. Hautamaki, and P. Fränti, "Knee Point Detection in BIC for Detecting the Number of Clusters," *Proc. 10th Int'l Conf. Advanced Concepts for Intelligent Vision Systems*, pp. 664-673, 2008.

[42] X. Tolsa, "Principal Values for the Cauchy Integral and Rectifiability," *Am. Math. Soc.*, vol. 128, no. 7, pp. 2111-2119, 2000.

[43] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a 'Kneedle' in a Haystack: Detecting Knee Points in System Behavior," *Proc. 31st Int'l Conf. Distributed Computing Systems Workshops (ICDCSW)*, pp. 166-171, 2011.

Ali Shahbazi received the BS and MS degrees in electrical engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2007 and 2010, respectively. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering at the University of Alberta. He is a member of STEAM lab working on software testing and machine learning. His research interests include software testing approaches in web-based systems, embedded systems, and mobile devices as well as security of the web based systems. He is a student member of the IEEE.

James Miller received the BSc and PhD degrees in computer science from the University of Strathclyde, Scotland. During this period, he was on the ESPRIT project GENEDIS on the production of a real-time stereovision system. Subsequently, he was at the United Kingdom's National Electronic Research Initiative on Pattern Recognition as a principal scientist, before returning to the University of Strathclyde to accept a lectureship, and subsequently a senior lectureship in computer science. Initially, during this period his research interests were in computer vision, and he was a co-investigator on the ESPRIT 2 project VIDIMUS. Since 1993, his research interests have been in web, software, and systems engineering. In 2000, he joined the Department of Electrical and Computer Engineering at the University of Alberta, Canada, as a full professor and in 2003 became an adjunct professor at the Department of Electrical and Computer Engineering at the University of Calgary. He has published more than 150 refereed journal and conference papers on web, software and systems engineering (see www.steam.ualberta.ca for details on recent directions). He recently served as the 2011 organizing chair for the IEEE International Symposium on Empirical Software Engineering and Measurement; and sits on the editorial board of the *Journal of Empirical Software Engineering*. He is a member of the IEEE.