ERROR CORRECTING ANALYSIS FOR TREE LANGUAGES

DAMIÁN LÓPEZ, JOSÉ M. SEMPERE and PEDRO GARCÍA

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46071, Valencia, Spain
E-mail: {dlopez,jsempere,pgarcia}@dsic.upv.es

To undertake a syntactic approach to a pattern recognition problem, it is necessary to have good grammatical models as well as good parsing algorithms that allow distorted samples to be classified. There are several methods that obtain, by taking two trees as input, the editing distance between them. In the following work, a polynomial time algorithm which processes the distance between a tree and a tree automaton is presented. This measure can be used in pattern recognition problems as an error model inside a syntactic classifier.

Keywords: Tree languages; editing distance; error correcting parsing; tree automata.

1. INTRODUCTION

The treatment of noisy or distorted samples has always been a problem in pattern recognition tasks, these distortions do usually appear as a side effect to the acquisition, preprocess or primitive extraction phases.

In order to avoid these problems, several techniques were developed to allow the correct classification of the noisy samples under a syntactic approach (mainly by using stochastic grammars and error-correcting parsers). These techniques were always applied on string languages.\(^1\),\(^2\),\(^3\),\(^4\),\(^9\),\(^10\) but these approximations have always lacked a method to classify distorted samples.

The algorithm by Lu and Fu,\(^6\) performs an error-correcting parsing on the basis of five-tree edition operations: label substitution of a leaf, insertion of a labeled node between another node and its direct predecessor, the insertion of a node to the left and to the right of a node, and the deletion of a node which has at most one successor.

In the same work a tree binary form is proposed. This normal form modifies the initial grammar to obtain a form whose productions have no more than two successors, and where, given a production, only one of the auxiliary symbols of its left-hand side can derive again with arity 2.

The first step in the algorithm is the application of the normal form both to the grammar productions and to the trees. A tree automaton is then built from the

© World Scientific Publishing Company
binary normal form grammar, and is then modified by adding productions according to the five edit operations in order to process the noisy samples.

The authors apply the error correcting method to a handwritten digit recognition task, obtaining good performance. Nevertheless, the time complexity of the algorithm is not shown, but the experimental measures that are taken point to a very important increase in the time necessary to process an error-correcting parsing in relation to a noncorrecting one.

In other approximations, there are several works that establish a distance between trees. Zhang and Shasha give an algorithm to calculate the distance between two trees in polynomial time,\textsuperscript{16} there are also works about the complexity of the comparison problem on nonordered trees.\textsuperscript{15,17} In the work by Oommen et al.,\textsuperscript{8} an algorithmic scheme is given to obtain any desired measure between trees. Furthermore, Oommen and Loke give a method to recognize a tree from one of its noisy subtrees.\textsuperscript{7}

All these papers allow the use of other pattern recognition techniques when the objects are modeled by a tree (\textit{k}-neighbours, clustering, etc.), but when syntactic techniques are applied, normally \( N \) languages arise, each corresponding to a class, and it is necessary to establish a distance between the trees and the languages to carry out the classification.

In this work, a polynomic time algorithm is proposed to calculate the distance between a tree and a given tree automaton. The paper begins with the definitions to be used along the work, then the error-correcting parser algorithm is proposed. A theorem proves its correctness, and then the complexity is calculated. The paper ends with the conclusions obtained.

2. DEFINITIONS AND NOTATION

Given \( V \) an alphabet and \( \mathbb{N} \) the set of natural numbers, let a \textit{ranked alphabet} be the association of \( V \) with a finite relation \( r \subseteq (V \times \mathbb{N}) \). \( V_n \) denotes the subset \( \{ \sigma \in V | (\sigma, n) \in r \} \).

Let \( V^T \) be the set of finite trees whose nodes are labeled with symbols in \( V \), where a tree is defined inductively as follows:

\[ V_0 \subseteq V^T \]
\[ \sigma(t_1, \ldots, t_n) \in V^T : \quad \forall t_1, \ldots, t_n \in V^T, \]
\[ \sigma \in V_n. \]

Let the root of a tree \( t \), denoted by \( \text{root}(t) \), be:

\[ \text{root}(a) = a : \quad \forall a \in V_0. \]
\[ \text{root}(\sigma(t_1, \ldots, t_n)) = \sigma : \quad \forall t_1, \ldots, t_n \in V^T, \]
\[ \sigma \in V_n. \]
Let the depth of a tree $t$, denoted by $\text{Depth}(t)$, be:

\[
\text{Depth}(a) = 0 : \quad \forall a \in V_0.
\]

\[
\text{Depth}(\sigma(t_1, \ldots, t_n)) = 1 + \max \{ \text{Depth}(t_i) \} \quad \forall t_i \in V^T, 1 \leq i \leq n; \quad \sigma \in V_n.
\]

Given a tree $t$, such that $\text{Depth}(t) \geq 1$, we define the successors of a tree $t$, denoted by $H^t$, as the following string:

\[
H^{\sigma(t_1, \ldots, t_n)} = (\text{root}(t_1), \ldots, \text{root}(t_n)) : \quad \forall t_1, \ldots, t_n \in V^T, \sigma \in V_n.
\]

Let the size of a tree $t$, denoted by $|t|$, be:

\[
|a| = 1 : \quad \forall a \in V_0.
\]

\[
|\sigma(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t_i| : \quad \forall t_1, \ldots, t_n \in V^T, \sigma \in V_n.
\]

A deterministic tree automaton is defined as the four-tuple $A = (Q, V, \delta, F)$ where $Q$ is a finite set of states; $V$ is a ranked alphabet; $F \subseteq Q$ is a set of final states and $\delta = (\delta_0, \ldots, \delta_m)$ is a finite set of transitions defined as:

\[
\delta_n : (V_n \times (Q \cup V_0)^n) \rightarrow Q \quad n = 1, \ldots, m,
\]

\[
\delta_0(a) = a \quad \forall a \in V_0.
\]

$\delta$ can be extended to operate on trees as follows:

\[
\delta : V^T \rightarrow Q \cup V_0
\]

\[
\delta(\sigma(t_1, \ldots, t_n)) = \delta_n(\sigma, \delta(t_1), \ldots, \delta(t_n)) \quad \text{if } n > 0,
\]

\[
\delta(a) = a \quad \forall a \in V_0.
\]

A tree $t \subseteq V^T$ is accepted by $A$ if $\delta(t) \in F$. The set of trees accepted by $A$ is defined as $L(A) = \{ t \in V^T | \delta(t) \in F \}$.

Given the state $q \in Q$, we define the ancestors of the state $q$, denoted by $\text{Ant}(q)$, as the set of strings:

\[
\text{Ant}(q) = \{ (p_1, \ldots, p_n) | p_i \in (Q \cup V_0) \land \delta_n(\sigma, p_1, \ldots, p_n) = q \}.
\]

In this work we shall use only one symbol ($\sigma$), to denote arities greater or equal than 1. Considering $\Sigma$ as the alphabet of symbols whose arity is 0, we shall deal with trees over $V = \{ \sigma \} \cup \Sigma$. These trees are named skeletons. From now on we shall use only skeletons, making no difference between them and trees.

Let $\sigma_i^t$ be the $i$th node of the tree $t$ according to a postorder enumeration, such that in the same level the nodes are ordered from left to right. Usually, if it is clear which tree $t$ is involved, we shall use $\sigma_i$ instead of $\sigma_i^t$.

Let the subtree from $t$ whose root is the node $\sigma_i$ be denoted by $S_i^t$. Let $H_i^t$ be the string formed by the successors of $\sigma_i$ in $t$.

By considering the edition costs over trees shown in Fig. 1, we define the distance between trees as the operations with minimum cost needed to transform one tree into the other.
3. TREE EDITING DISTANCE

To calculate the distance between a tree \( t \) to a tree automaton \( A \), we propose an algorithm which explores the tree, calculating, \( \forall \sigma_i \in t \), the cost to reduce the tree \( S^t_i \) to every state of the automaton.

In this way, given a node of a tree \( \sigma_i \) such that \( S^t_i = \sigma(t_1, t_2, \ldots, t_n) \) and a state \( q \in Q \), the strings involved into the comparison are: the ancestors of each state of the automaton \( \text{Ant}(q) \), \( q \in Q \), and the sequences \( \tilde{H}^t_i = (h_1, h_2, \ldots, h_n) \) where \( h_i \in \{t_i\} \times Q \) if \( \text{Depth}(t_i) \geq 1 \) and \( h_i = (t_i, t_i) \) otherwise. Intuitively \( \tilde{H}^t_i \) extends the notion of \( H^t_i \) to chains of states instead of tree nodes. This measure is stored in a matrix \( (D_A) \) indexed by the states and the tree nodes.

To carry out this calculation, it is necessary to extend the edit operations to take into account every possibility. These operations are shown in Fig. 2.

The method visits the tree nodes in postorder, and in each node carries out string distance calculations. Given a generic string edition algorithm \( D_C \), (e.g. Ref. 11), let \( \overline{D_C} : (V^T \times (Q \cup V^T))^* \times (Q \cup V_0)^* \rightarrow \mathbb{N} \) be the algorithm which maintains the same algorithmic scheme, but using the defined editing operations. This algorithm gives the minimum distance by using the edition operations on trees previously defined (Fig. 2).
Now, we can establish the distance between a tree \( t = \sigma(t_1, t_2, \ldots, t_k) \) and a state \( q \in Q \), with the following expression:

\[
D(t, q) = \begin{cases} 
\text{Min} \{ D_C(\hat{H}^t, x) | x \in \text{Ant}(q) \} & \text{if } \text{Depth}(t) = 1 \\
\text{Min} \left\{ D_C(((t_1, q_{i_1}), (t_2, q_{i_2}), \ldots, (t_k, q_{i_k})), x) + \sum_{j=1}^{k} D(t_j, q_{i_j}) \right\} & \text{otherwise}
\end{cases}
\]

After the edition operations have been defined, let the distance from a tree \( t \) to a tree automaton \( A \) be the operations with minimum cost necessary to allow the tree automaton to accept the tree.

\[
D(t, A) = \text{Min}\{ D(t, q) | q \in F \}
\]

3.1. Algorithm Description

In the proposed method, the insertion, deletion and substitution costs vary depending on which node is being analyzed, and these can be calculated dynamically. This is the main feature of this error-correcting parsing algorithm.

Under a dynamic programming scheme and using the postorder strategy, when a node \( \sigma_i, S^i = \sigma(t_1, t_2, \ldots, t_k) \) is going to be analyzed, every distance between \( S^i \) and the states of the automaton has already been calculated and stored in \( D_A \), avoiding thereby calculations that were carried out previously. A scheme of the algorithm is shown in Fig. 3.

---

**Input:** A finite tree automaton \( A \).
A tree \( t \).

**Output:** Distance from the tree \( t \) to the automaton \( A \).

**Method:**

```plaintext
/* Initialization. */
\forall \sigma_i \in t
\forall q \in Q
D_A[\sigma_i, q] = \infty

\forall \sigma_i \in t /* postorder enumeration */
\forall q \in Q
D_A[\sigma_i, q] = \text{Min}(D_A[\sigma_i, q], \text{Min}_{q \in \text{Ant}(q)} (D_C(S^i, x)))
```

**EndMethod.**

---

Fig. 3. Tree to automaton distance algorithm.
Example 1. Given the tree from Fig. 4 and the tree automaton from Fig. 5, a sketch of the algorithm would be:

\[
\begin{align*}
\text{Ant}(q_0) &= \{ (q_1, q_2) \} \\
\text{Ant}(q_1) &= \{ (aq_1 b), (ab) \} \\
\text{Ant}(q_2) &= \{ (aq_2), (a) \}
\end{align*}
\]

\[
\begin{align*}
D_A[\sigma_1, q_0] &= \widehat{D_C}(\langle (a, a) \rangle, q_1 q_2) = 6 \\
D_A[\sigma_1, q_1] &= \min \begin{cases} \\
\quad D_C(\langle (a, a) \rangle, aq_1 b) = 3 \\
\quad D_C(\langle (a, a) \rangle, ab) = 1 \\
\end{cases} \\
D_A[\sigma_1, q_2] &= \min \begin{cases} \\
\quad D_C(\langle (a, a) \rangle, aq_2) = 3 \\
\quad D_C(\langle (a, a) \rangle, a) = 0 \\
\end{cases} \\
D_A[\sigma_2, q_0] &= \min \{ \widehat{D_C}(\langle (a, a)(t_1, q) \rangle, q_1 q_2) | q \in Q \} = 3 \\
D_A[\sigma_2, q_1] &= \min \begin{cases} \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_1, q) \rangle, aq_1 b) | q \in Q \} = 2 \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_1, q) \rangle, ab) | q \in Q \} = 3 \\
\end{cases} \\
D_A[\sigma_2, q_2] &= \min \begin{cases} \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_1, q) \rangle, aq_2) | q \in Q \} = 0 \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_1, q) \rangle, a) | q \in Q \} = 2 \\
\end{cases} \\
D_A[\sigma_3, q_0] &= \min \{ \widehat{D_C}(\langle (a, a)(t_2, q)(b, b) \rangle, q_1 q_2) | q \in Q \} = 4 \\
D_A[\sigma_3, q_1] &= \min \begin{cases} \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_2, q)(b, b) \rangle, aq_1 b) | q \in Q \} = 2 \\
\quad \min \{ \widehat{D_C}(\langle (a, a)(t_2, q)(b, b) \rangle, ab) | q \in Q \} = 4 \\
\end{cases}
\]

\[
\begin{align*}
\delta(\sigma, q_1, q_2) &= q_0 \\
\delta(\sigma, a, q_1, b) &= q_1 \\
\delta(\sigma, a, b) &= q_1 \\
\delta(\sigma, a) &= q_2 \\
F &= \{ q_0 \}
\end{align*}
\]

Fig. 4. A tree example and the postorder enumeration of its internal nodes.

Fig. 5. A tree automaton example.
\[ D_A[\sigma_3, q_2] = \min \left\{ \begin{array}{l} \min \{D_C((a, a)(t_2, q)(b, b), aq_2)|q \in Q\} = 1 \\ \min \{D_C((a, a)(t_2, q)(b, b), a)|q \in Q\} = 5 \end{array} \right. \]

Nodes \( \sigma_4 \) and \( \sigma_5 \) are equivalent to \( \sigma_1 \) and \( \sigma_2 \) respectively, so that finally:

\[ D_A[\sigma_6, q_0] = \min \{D_C((t_3, p)(t_4, q), q_1q_2)|p, q \in Q\} = 2 \]
\[ D_A[\sigma_6, q_1] = \min \left\{ \begin{array}{l} \min \{D_C((t_3, p)(t_4, q), aq_1b)|p, q \in Q\} = 6 \\ \min \{D_C((t_3, p)(t_4, q), ab)|p, q \in Q\} = 13 \end{array} \right. \]
\[ D_A[\sigma_6, q_2] = \min \left\{ \begin{array}{l} \min \{D_C((t_3, p)(t_4, q), aq_2)|p, q \in Q\} = 8 \\ \min \{D_C((t_3, p)(t_4, q), a)|p, q \in Q\} = 12 \end{array} \right. \]

The values of the matrix \( D_A \) would be those shown in Fig. 6:

<table>
<thead>
<tr>
<th>( D_A )</th>
<th>( \sigma_1 )</th>
<th>( \sigma_2 )</th>
<th>( \sigma_3 )</th>
<th>( \sigma_4 )</th>
<th>( \sigma_5 )</th>
<th>( \sigma_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Fig. 6. Distances matrix. The value \( D_A[\sigma_6, q_0] \) shows the distance between the tree automaton in Fig. 5 and the tree in Fig. 4.

3.2. Comparison Between Error-Correcting Parsers

In Example 2 of the paper by Lu and Fu, the behavior of the algorithm when applied to the recognition of characters is shown.

The authors build an expanded tree automaton from the standard character “E” shown in Fig. 7. The number of productions of the automaton is 68, and the process performed to parse another causally written character “E” (shown in Fig. 8) takes eight main steps, as shown in Fig. 7 of the referred paper.

Fig. 7. A standard character “E” and its tree representation.
To compare both methods, since we use skeletons instead of binarized trees, it is necessary to transform the trees that Lu and Fu binarize in order to obtain a skeleton. An easy way to do so, can be inductively explained as follows:

$$Sk(a) = a : \forall a \in V_0.$$  
$$Sk(\alpha(t_1, \ldots, t_n)) = \sigma(\alpha, Sk(t_1), \ldots, Sk(t_n)) : \forall t_1, \ldots, t_n \in V^T, \alpha \in V_n, \sigma \notin V.$$  

This process, for each internal node labeled with $\alpha \in V$, creates a new leftmost leaf, labeling it with $\alpha$, therefore, every symbol in $V$ becomes a symbol in $V_0$, and all the internal nodes of the tree are labeled by a new symbol $\sigma \notin V$.

Once the same samples used by Lu and Fu are skeletized (Fig. 9), let the productions of the automaton which accept just the standard “E” be the following:

$$\delta(\sigma, d) = q_1$$  
$$\delta(\sigma, b, q_1) = q_2$$  
$$\delta(\sigma, b, q_2, d) = q_3$$  
$$\delta(\sigma, $, q_3, d) = q_4 \in F$$  

The parser scheme is shown in Fig. 10.

We can conclude that the method we propose need no extension of the automata, dealing with smaller ones, and since no normal form is needed, the number of steps to carry out the parsing task is also lower than the number of steps that is needed by Lu and Fu’s algorithm.
3.3. Correctness of the Algorithm

**Theorem 1.** Given $t \in V^T$ and $A = (Q, \Sigma, \delta, F)$, the proposed algorithm obtains the minimum distance from $t$ to the automaton $A$ (minimum cost necessary to make $A$ accept $t$).

**Proof.** We shall prove the statement by induction on the depth of the subtrees of $t$.

- **Induction Basis:**
  
  Suppose $\text{Depth}(S_i^1) = 1$.
  
  In this case, the distance from $S_i^1$ to each state $q \in Q$ coincides with the minimum of the set $\{D_C(H_i^1, x) | x \in \text{Ant}(q) \}$. When $\text{Depth}(t) = 1$, then, the distance from the tree $t$ to the automaton $A$ will be $\text{Min}\{D_C(H^t, x) | x \in \text{Ant}(q) \land q \in F \}$.

- **Induction Hypothesis:**
  
  Suppose $\text{Depth}(S_i^j) \leq n$.
  
  Given $S_i^j||\text{Depth}(S_i^j) \leq n$, the algorithm calculates the minimum distance from each subtree $S_i^j||\text{Depth}(S_i^j) \leq n$ to each state $q \in Q$.

- **Induction Step:**
  
  Suppose $\text{Depth}(S_i^j) = n + 1$.
  
  The postorder enumeration strategy assures that, when a subtree $S_i^j = \sigma(t_1, t_2, \ldots, t_n)$, $\text{Depth}(S_i^j) = n + 1$, is going to be analyzed, $D(S_i^j, q)$ has already been calculated. By **Induction Hypotheses**, $D(S_i^j, q)$ is the minimum edition cost to reduce $S_i^j$ to every state $q \in Q$.
  
  Therefore, given a tree $t = \sigma(t_1, t_2, \ldots, t_k)$, and given the set:

  \[
  \left\{ D_C((t_1, q_{i_1}), (t_2, q_{i_2}), \ldots, (t_k, q_{i_k}), x) + \sum_{j=1}^{k} D(t_j, q_{i_j}) \mid x \in \text{Ant}(q), \langle (t_1, q_{i_1}), (t_2, q_{i_2}), \ldots, (t_k, q_{i_k}) \rangle \in \widehat{H^t} \right\}
  \]
since \( \widehat{D}_C \) calculates the minimum distance between two strings, and the summatory is minimum because each of its terms \( D(t_j, q_{ij}) \) is also minimum, the minimum value of the set coincides with \( D(t, q) \), and therefore, when \( S_i^t = t \), \( \min \{D(t, q) | q \in F \} \) is the minimum distance from \( t \) to the automaton \( A \).

\[ \square \]

### 3.4. Algorithm Complexity

Given a tree \( t \) and a tree automaton \( A = (Q, \Sigma, \delta, F) \) (we consider \( |A| = p \) and \( |t| = m \)), the algorithm works out the editing distance between the sets of strings \( H_i^t \) and \( \text{Ant}(q) \), \( \forall q \in Q \).

Let the time complexity of a generic string distance algorithm \( D_C \) be \( \mathcal{O}(n) = \mathcal{P}(n) \), \( n \) being the length of the longest string.\(^{11} \) Given the algorithm \( \widehat{D}_C \), the algorithmic structure of \( D_C \) is maintained, and every editing operation is bounded by \( \mathcal{O}(m) = m \), the editing cost of the strings is \( \mathcal{O}(m, p) = \mathcal{P}^\prime(\max(m, p)) \), where \( \mathcal{P}^\prime \) is a polynomial function.

Considering that this operation is carried out \( p \) times in each internal node of the tree, the time complexity of the algorithm is given by:

\[
\mathcal{O}(p, m) = \mathcal{P}''(\mathcal{P}^\prime(\max(m, p)) \times p \times m)
\]

where \( \mathcal{P}'' \) is a polynomial function.

The space complexity is bounded by the matrix \( D_A \) which stores the minimum distance from each internal node to each state of the automaton. It is then given by: \( \mathcal{O}(p \times m) \).

### 4. CONCLUSIONS

The tree representation allows more complex patterns to be dealt with more effectively. Therefore, when syntactic techniques are applied to pattern recognition problems, this representation implies an improvement over the classic string representation.

There exist several methods to measure the distance between trees\(^ {7,8,12,16} \) all of which allow the application of geometric techniques to pattern recognition problems. The proposed algorithm offers the advantage to carry out an error-correcting parsing over tree automata, allowing the use of tree language inference methods in this area by including the algorithm as an error correcting model.

Considering the other error-correcting parser commented in the introduction,\(^6 \) the proposed algorithm does not need a normal form to be applied either to the automaton nor to the trees to be classified. Furthermore, it does not need to add new productions to the automaton. Although the authors\(^6 \) do not establish the complexity, the comparison between both methods shows us that the method by Lu and Fu deals with greater automata and trees than the method which is here proposed.

Finally, this method can be easily modified to consider internal nodes with different labels, that is, to work on trees instead of skeletons.
ACKNOWLEDGMENT
Damián López would like to thank Prof. B. J. Oommen for his helpful comments.

REFERENCES
Damián López graduated in computer science in 1995 at the Universidad Politécnica de Valencia (UPV), Spain. He joined the Departamento de Sistemas Informáticos y Computación in 1996, and since then he teaches at the School of Industrial Engineering (UPV). His current fields of interest include multidimensional formal languages and their application in syntactic pattern recognition tasks. He is also a member of the Spanish Association of Pattern Recognition and Image Analysis (AERFAI), affiliated society to the IAPR.

José M. Sempere graduated in computer science in 1992 at the Universidad Politécnica de Valencia (UPV), Spain. In 1989, he joined the Departamento de Sistemas Informáticos y Computación (DSIC) of the UPV and since then he teaches in the Computer Science Faculty. His current fields of interest include computational learning theory, formal languages, and pattern recognition. He is also a member of the EATCS.

Pedro García graduated in physics in 1976 at the Universidad de Valencia and received the Ph.D. in computer science in 1988 at the Universidad Politécnica de Valencia (UPV), Spain. In 1987, he joined the Departamento de Sistemas Informáticos y Computación (DSIC) of the UPV and since then he teaches in the Computer Science Faculty. His current fields of interest include automata theory and formal languages.