Semantic tree operations

D. Gueorguiev 1/6/2021

Notation:
$T^{(m)}$ – an m-ary tree
$\mathcal{V}(T)$ - the set of all nodes of T
$P(v,T)$ – path from root of T to node v
(k,v) – tree tuple of T
k – tree factor, an integer
$\{\dot{k}_1,\dot{k}_2,\dots,\dot{k}_h\}$ - tuple of primitive factors
$\dot{k}$ – primitive factor, an integer
$\mathcal{K}(T)$ – the set of all primitive factor tuples of T
$\mathcal{A}(T)$ – the set of all arcs of T
$\mathcal{W}(T)$ – the set of all weights of T

Summary and definitions

We are considering m-ary tree $T^{(m)}$ (abbreviated with T from now on), which is a tree in which each node has at most m children. Let us introduce the *tree tuple* (k,v) where v represents a *tree node* which is a semantic particle or a *subtree* of nodes (i.e. semantic particles). We will denote by $\mathcal{V}(T)$ the set of all nodes which belong to the tree T.
Here k is a *tree factor* which encodes uniquely the position of the node or the root of the subtree v in the parent tree T. More precisely, the factor k encodes uniquely the path $P(v,T)$ from the root of the tree to the node v or the root of subtree associated with k. Each tree factor k representing a node other than the root of T can be decomposed into a *tuple* t of *primitive factors* $\{\dot{k}_1,\dot{k}_2,\dots,\dot{k}_h\}$ where h is smaller or equal to the height of the tree. Here with dot-accented $\dot{k}$ we denote a primitive factor i.e. a factor which cannot be represented by any other combination of primitive factors.
We will denote with $\mathcal{K}(T)$ the set of all tuples of primitive factors associated with tree T. The position in t of each of those primitive factors and their value encodes an *arc* from the set of arcs forming the path $P(v,T)$.

Definition: *Arc* of semantic tree
Let us define a tree with N nodes as:

$$T = \sum_{i=0}^{N}(k_i,v_i) \qquad (1)$$

We denote with t_i the tuple of the primitive factors for k_i:

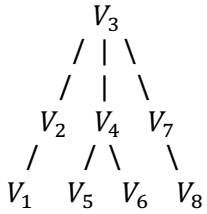
$$t_i = \{\dot{k}_{1,i},\dot{k}_{2,i},\dots,\dot{k}_{h,i}\}. \qquad (2)$$

Every arc of T is represented by a tuple $t = \{\dot{k}_1,\dot{k}_2,\dots,\dot{k}_l\}$ such that $t \in \mathcal{K}(T)$. However not every tuple $t \in \mathcal{K}(T)$ represents an arc - the tuple $\{\dot{k}_0\}$ associated with the root of T does not correspond to an arc of T. We will denote with $\mathcal{A}(T)$ the set of all arcs of T. Obviously, $|\mathcal{A}(T)| = |\mathcal{K}(T)|\text{-}1 = |\mathcal{V}(T)|\text{-}1$.

If we substitute (2) in (1) we come up with the following simplified notation for T represented in terms of its primitive factors which will be used throughout this discussion:

$T = \sum_{i=0}^{N}(\dot{k}_{1,i}, \dot{k}_{2,i}, \ldots, \dot{k}_{h,i}, v_i)$ (3)

Example



The root node V_3 is given with the tuple (k_0, v_3). Its children are given with (k_1, v_2), (k_2, v_4), and (k_3, v_7). Their children are accordingly (k_4, v_1), (k_5, v_5), (k_6, v_6), and (k_7, v_8). In a later paragraph we will introduce rules which will allow us to show that the following expansions take place:

$k_1 = \dot{k}_1\dot{k}_0$
$k_2 = \dot{k}_2\dot{k}_0$
$k_3 = \dot{k}_3\dot{k}_0$
$k_4 = \dot{k}_1\dot{k}_1$
$k_5 = \dot{k}_2\dot{k}_1$
$k_6 = \dot{k}_2\dot{k}_2$
$k_7 = \dot{k}_3\dot{k}_1$

Thus, the tree T can be written as:
T = $(\dot{k}_0, v_3) + (\dot{k}_1\dot{k}_0, v_2) + (\dot{k}_2\dot{k}_0, v_4) + (\dot{k}_3\dot{k}_0, v_7) + (\dot{k}_1\dot{k}_1, v_1) + (\dot{k}_2\dot{k}_1, v_5) + (\dot{k}_2\dot{k}_2, v_6) + (\dot{k}_3\dot{k}_1, v_8)$.

Notice that for the root of the tree we always have:

$k_0 = \dot{k}_0$ (4)

Definition: *Matching arcs of trees*
Let us consider two trees represented as:

$T = \sum_{i=0}^{N}(k_i, v_i)$ and $T^* = \sum_{i=0}^{N}(k_i^*, v_i^*)$

If for some i and j we have $k_i = k_j^*$ then the arcs which correspond to k_i and k_j^* are said to be *matching*.

Definition: *Weighted semantic tree*
Let there exists a function f which maps each tree factor k_i to a real number which is the weight corresponding the arc associated with the specified tuple factor. So, the number $f(k_i)$ associated with the arc corresponding to $k_i > 0$ is the arc weight. With $\mathcal{W}(T)$ we will denote the set of all $f(k_i), k_i > 0$ where $k_i \in \mathcal{K}(T)$.

Definition: *Weight of a tree path*
Let us evaluate the weight of the path P_ℓ from root to node $v_l, l > 0$. Let k_l be the factor associated with v_l and $k_l = \{\dot{k}_{1,l}, \dot{k}_{2,l}, \ldots, \dot{k}_{h,l}\}$. Then the weight of the path P_ℓ is given with:

$$f(P_\ell) = \prod_{i=1}^{h} f\left(\prod_{j=1}^{i} \dot{k}_{j,l}\right) \qquad (5)$$

Definition: *Semantic significance vector of a semantic tree*
In various settings we will use a weight vector instead of scalar weight with semantic trees and semantic paths. Thus, the vector-valued function $\vec{f}$ will map each tree factor $k_i \in \mathcal{K}(T)$ to an element $\vec{w}$ of some vector space $\mathbb{R}^n$. We will refer to each element $\vec{w}$ of $\mathbb{R}^n$ mapped to a tree arc or a path as *semantic significance vector*.
//TODO

Semantic subtree expansion and node comparison

The following operations are defined for tree factors:

Multiplication operation for semantic tuple factors

One possible implementation for the primitive factors $\dot{k}_1, \dot{k}_2, \ldots \dot{k}_m$ is to define them as the m digits greater than 0 of $(m+1)$-nary number system such that $0 = \dot{k}_0 < \dot{k}_1 < \dot{k}_2 < \ldots < \dot{k}_m$. We define an operation `*` denoting digit concatenation $\dot{k}_i * \dot{k}_j = (m+1)\dot{k}_i + \dot{k}_j$. Obviously,
$\dot{k}_i * \dot{k}_j > \dot{k}_i$ for any pair $i, j = 1..m$
Note that the latter implies that

$\dot{k}_{i_1} * \dot{k}_{i_2} * \ldots * \dot{k}_{i_n} > \dot{k}_{j_1} * \dot{k}_{j_2} * \ldots * \dot{k}_{j_{n-1}}$ for any tuple where $i_p, j_q = 1..m, p = 1..n, q = 1..n-1$

$$\left(\dot{k}_i, (\dot{k}_j, v_j)\right) = (\dot{k}_i * \dot{k}_j, v_j)$$

Encoding a complete m-ary tree T of height h with the algebraic notation above:
$T = (\dot{k}_0, v_0^0)$. Further we will assume that $\dot{k}_0 = 0$.
$v_0^0 = (\dot{k}_0, v_0^1) + (\dot{k}_1, v_1^1) + (\dot{k}_2, v_2^1) + \ldots + (\dot{k}_m, v_m^1)$

In general, we have:
$v_q^p = \left(\dot{k}_0, v_{(q-1)m}^{p+1}\right) + \left(\dot{k}_1, v_{(q-1)m+1}^{p+1}\right) + \left(\dot{k}_2, v_{(q-1)m+2}^{p+1}\right) + \ldots + \left(\dot{k}_m, v_{qm}^{p+1}\right)$
where $q = 1..m^h, p = 1..h$
Obviously, we have at most $\frac{(m^{h+1}-1)}{m-1}$ distinct terms v_q^p which represent nodes i.e. semantic values.

Tuple factor comparison operator

The expression for the tree also can be written as:

$T = \sum_{i=0}^{N}(k_i, v_i)$ where $N \leq \frac{(m^{h+1}-1)}{m-1}$ and k_i are the *node factors* given with $k_i = \dot{k}_{i_1} * \dot{k}_{i_2} * \ldots * \dot{k}_{i_n}; n \leq h$. The node values v_i are the values v_q^p ordered in increasing order of k_i. This order corresponds to *level order traversal* of the m-ary tree. Note that with appropriately defined comparison operation `<` we can model different ways of traversing the m-ary tree. For instance, if we define `<` as the comparison for the values of $(m+1-\dot{k}_{i_1})m^{n-1} + (m+1-\dot{k}_{i_2})m^{n-2} + \cdots + (m+1-\dot{k}_{i_n})$ we will have ordering which corresponds to the *preorder traversal* of the tree.

Example

Peter is Dimitar's son.

Dimitar's son has a friend in the neighborhood and his friend's name is James.

⇨ *James is Peter's friend*

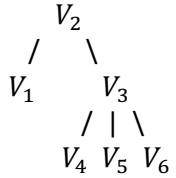
Peter is the son of Dimitar.

V_1 V_2 V_3 V_4 V_5 V_6

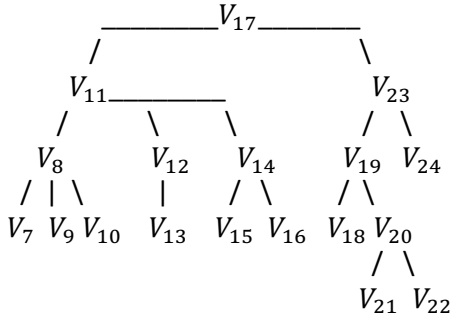
The son of Dimitar has a friend in the neighborhood and the name of his friend is James.

V_7 V_8 V_9 V_{10} V_{11} V_{12} V_{13} V_{14} V_{15} V_{16} V_{17} V_{18} V_{19} V_{20} V_{21} V_{22} V_{23} V_{24}

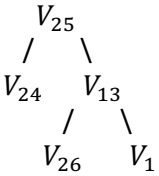
T_1



T_2



T_3



Expressing T_1 with the algebraic notation discussed earlier:

$T_1 = (k_0, v_2) + (k_1, v_1) + \big(k_2, \big((k_0, v_3) + (k_1, v_4) + (k_2, v_5) + (k_3, v_6)\big)\big)$

which is expanded to:

$T_1 = (k_0, v_2) + (k_1, v_1) + (k_2k_0, v_3) + (k_2k_1, v_4) + (k_2k_2, v_5) + (k_2k_3, v_6)$

Expressing T_2 with the algebraic notation yields:

$$T_2 = (k_0, v_{17}) + \Big(k_1, \Big((k_0, v_{11}) + \Big(k_1, \big((k_0, v_8) + (k_1, v_7) + (k_2, v_9) + (k_3, v_{10})\big)\Big) + \Big(k_2, \big((k_0, v_{12}) + (k_1, v_{13})\big)\Big) + \Big(k_3, \big((k_0, v_{14}) + (k_1, v_{15}) + (k_2, v_{16})\big)\Big)\Big)\Big) + \Big(k_2, \Big((k_0, v_{23}) + \Big(k_1, \Big((k_0, v_{19}) + (k_1, v_{18}) + \Big(k_2, \big((k_0, v_{20}) + (k_1, v_{21}) + (k_2, v_{22})\big)\Big)\Big)\Big) + (k_2, v_{24})\Big)\Big)$$

which is expanded to:

$$\begin{aligned} T_2 = {} & (k_0, v_{17}) + (k_1k_0, v_{11}) + (k_1k_1k_0, v_8) + (k_1k_1k_1, v_7) + (k_1k_1k_2, v_9) + (k_1k_1k_3, v_{10}) \\ & + (k_1k_2k_0, v_{12}) + (k_1k_2k_1, v_{13}) + (k_1k_3k_0, v_{14}) + (k_1k_3k_1, v_{15}) + (k_1k_3k_2, v_{16}) \\ & + (k_2k_0, v_{23}) + (k_2k_1k_0, v_{19}) + (k_2k_1k_1, v_{18}) + (k_2k_1k_2k_0, v_{20}) + (k_2k_1k_2k_1, v_{21}) \\ & + (k_2k_1k_2k_2, v_{22}) + (k_2k_2, v_{24}) \end{aligned}$$

Semantic Tree Difference

Definition: *Semantic Tree Difference* – a metric (real valued function) which maps two semantic trees of the same kind into a real value.

Let us have two trees represented as:

$$T = \sum_{i=0}^{N}(k_i, v_i) \text{ and } T^* = \sum_{i=0}^{N}(k_i^*, v_i) \quad (6)$$

Here $v_i, i = 1..N$ denote the nodes of the two trees (not subtrees) which are semantic particles with signatures $ssig(v_i)$. The sequences k_i and k_i^* denote the sequences of tuple factors which encode the position of each node v_i in each of the two trees.

Let us assume that the two trees are weighted so that for each tree there is weight function which maps each tuple factor to a real number which is the weight corresponding the arc associated with the specified tuple factor. Let us denote by f and f^* the two weight functions corresponding to T and T^*. Generally, $f(k_i) \neq f^*(k_i)$. If $f \equiv f^*$ then T and T^* have the same weights on their matching arcs.

We have the same set of semantic particles but they are arranged differently in two trees. We would like to define metric how different are the two trees.

We will consider the following special cases – the trees which we would like to obtain a
Metric for calculating Semantic Property Tree Difference will be two kinds:

- Semantic Property Trees
- Semantic Particle Trees

First, we need to elaborate on a metric defining a semantic difference between two semantic property trees

Semantic Difference between Semantic Property Trees

Definition: *Semantic difference between semantic properties*
Let us consider two semantic properties P_1 and P_2 given with their semantic signature matrices $ssig(P_1) = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_m]$ and $ssig(P_2) = [\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n]$ where the column vectors $\vec{a}_i, i = 1..m$ and $\vec{b}_j, j = 1..n$ are the semantic aspects of P_1 and P_2 accordingly. Without loss of generality we will assume that $m \leq n$. Then the semantic difference between the properties P_1 and P_2 is given with

$$sdiff(P_1, P_2) = \sum_{i,j} \left| \frac{\mathfrak{m}_i}{E_i} \vec{a}_i - \frac{\mathfrak{m}_j}{E_j} \vec{b}_j \right| + \sum_k \frac{\mathfrak{m}_k}{E_k} \left| \vec{b}_k \right| \qquad (7)$$

where $\mathfrak{m}_i$ and E_i denote the semantic mass and energy of the aspect $\vec{a}_i$ of P_1 while $\mathfrak{m}_j$ and E_j denote the corresponding quantities for the $\vec{b}_j$ of P_2. The summation in the first summation term occurs over all (i, j) pairs such that for a given i the index j is selected as that aspect index from P_2 which yields the smallest value of $\left| \frac{\mathfrak{m}_i}{E_i} \vec{a}_i - \frac{\mathfrak{m}_j}{E_j} \vec{b}_j \right|$. The aspect indices of P_1 are ordered by decreasing $\frac{\mathfrak{m}_i}{E_i}$ values so we start with the aspect having the highest ratio of $\frac{\mathfrak{m}_i}{E_i}$ and attempt to match it with similar one from P_2.
The second summation term in (7) represents the unmatched aspects from P_2 which obviously are $n - m$.

Definition: *Semantic difference between semantic property trees*
Let us consider the property trees represented as:

$T = \sum_{i=0}^{N} (k_i, p_i)$ and $T^* = \sum_{i=0}^{N} (k_i^*, p_i)$

Both trees share the same property set $p_i, i = 1..N$. As before we will denote by f and f^* the two weight functions corresponding to T and T^*.
We would like to evaluate how much different semantically are the two trees
//TODO