A New tree similarity measuring methodology and its application to ontology comparison

Xue, Y.; Wang, C.; Ghenniwa, H.; Shen, W.

Publisher's version / Version de l'éditeur:

12th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008) [Proceedings], pp. 258-263, 2008-04-16

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=ee1ef45f-a36d-4604-858c-d457cec03459
https://publications-cnrc.canada.ca/fra/voir/objet/?id=ee1ef45f-a36d-4604-858c-d457cec03459

A new tree similarity measuring methodology and its application to ontology comparison

NRCC-50280

Xue, Y.; Wang, C.; Ghenniwa, H.H.; Shen, W.

A version of this document is published in / Une version de ce document se trouve dans:
Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008), Xi'an, P.R. China, April 16-18, 2008, pp. 258-263

A New Tree Similarity Measuring Methodology and its Application to Ontology Comparison

Yunjiao Xue, Chun Wang, Hamada H. Ghenniwa, Weiming Shen
Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON, Canada
yxue24@uwo.ca, cwang28@uwo.ca, hghenniwa@eng.uwo.ca, wshen@uwo.ca

Abstract

This paper extends the classical tree similarity measuring methodology and proposes a definition for cost of tree transformation operations based on the importance of each concept in the entire concept structure and similarity between individual concepts in a knowledge context. We apply the proposed methodology to ontology comparison where different ontologies for the same domain are represented as trees and their similarity is required to be measured. We show that the proposed methodology can facilitate the initiation of ontology integration and ontology trust evaluation.

Keywords: Tree Similarity, Measuring, Transformation Cost, Ontology Comparison, Collaborative Design.

1. Introduction

An ontology specifies a conceptualization of a domain in terms of concepts and their relationships [9]. Ontology can create an agreed-upon vocabulary for sharing knowledge, exchanging information, and eliminating ambiguity. It plays a very important role in the area of Collaborative Design [1].

In practice, it is not common that a shared ontology be provided for a specific domain. Contrarily, usually different communities build their own ontologies that are heterogeneous in terms of structure, syntax, and semantics, even committing to the same conceptualization of that domain. Therefore, ontology integration [6] is developed to address this heterogeneity.

In some cases the ontologies need to be compared to support initialization of ontology integration and ontology trust evaluation. An ontology can be viewed as a knowledge structure, and one commonly used form is that of a tree structure. Much os the research on comparing trees uses editing cost from one tree to another to measure the similarity of two trees [2]. Classical methods focus on structural and geometrical characteristics of trees, mainly considering the number of nodes affected by editing operations [3, 7]. However, in a knowledge context where trees are used to model concept structures, in addition to structural characteristics of the trees, more attention must be paid to concepts represented by the internal tree nodes. Therefore, besides the number of edited nodes, positions and conceptual similarities of the affected nodes also have to be considered.

The similarity of two concept structures is based on the similarity of their member concepts. The similarity of two individual concepts can be relatively easily estimated by domain experts. As an example, based on common sense, concepts "*People*" and "*Human*" are often regarded as referring to the same meaning, i.e. their similarity degree is 1. On the other hand, concept "*Faculty*" is not always exactly referring to the same thing as "*Professor*". Roughly speaking, a similarity degree can be assigned to these two concepts, say, 0.9. Some researches have also proposed various methods of determining conceptual similarity between individual concepts in a knowledge context [4, 5].

Determining the similarity of various structures containing many concepts is fairly complicated. For instance, given the following three trees in Figure 1 (which are modeling the concept structure about university domain and are developed by different people) where relationships between concepts are identical ("*part-of*" in this example) and a list describing the similarity of individual concept pairs (e.g. sim(*People*, *Human*) = 1 and sim(*Faculty*, *Professor*) = 0.9) which can be provided by domain experts, how can we determine what extent they are similar to each other and which two are more similar?

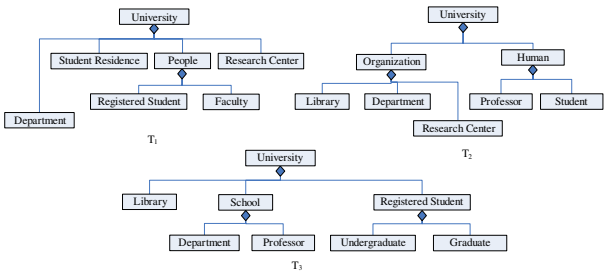

Figure 1. An example of multiple trees for one domain.

Our work extends classical tree editing operations. We propose four types of transformation operations which can map one concept tree into another, and provide definitions for the cost of each operation considering the number of affected nodes, the scale of the node set, the conceptual significance of affected nodes, and the conceptual similarity of the node pairs (each node representing one concept) in a knowledge

context. The degree of tree similarity is measured according to the tree transformation cost. This methodology can be applied to ontology comparison to support ontology integration in cases where different ontologies for the same domain can be represented as trees.

The rest of this paper is organized as follows. In section 2 we analyze previous work in related topics, and then in section 3 we present basic definitions of our work. Section 4 and 5 discuss tree transformation operations and their costs in detail. A case study in ontology comparison is discussed in section 6. Finally section 7 concludes the whole paper and proposes our future work.

2. Related Work

Research on comparing tree structures has a long history in many fields. Tree patter matching is one of the often used methods. For example, some researches explored the algorithm of matching pattern discovery in XML query [13, 14] whereas they did not focus on the cost of matching. Another domain of using tree pattern matching is compiling where matching cost is defined through tree-rewriting rules and instruction types [15].

Some researches have discussed the topic of tree editing and its cost (edit distance) [7, 8, 3]. However, these researches are mainly focused on finding matches based on pure structure or geometry perspective without considering the conceptual semantics of the tree nodes in a knowledge context.

Maedche et al conducted in-depth research of the similarity between ontologies [16]. In their research context, an ontology has a tree structure that is modeling a concept taxonomy. A methodology was developed to measure the similarity between ontologies based on the notions of *lexicon*, *reference functions*, and *semantic cotopy*. This method is based on an assumption that the same terms are used in different ontologies for concepts but their relative positions may vary. However, in many real ontologies different terms will be adopted to construct the concept taxonomies, although some of them have similar semantics. In these cases computing taxonomic overlap is not fully applicable and lexical level comparison becomes almost inapplicable. Furthermore, this research did not take the structural characteristics of trees into consideration.

Li et al conducted similar research on measuring the similarity of ontology models (represented as tree) based on tree structure mapping [17]. They proposed a mapping method that combines the similarity of the inner structure of concepts in different ontologies and the language similarity of concepts. The similarity of concepts is computed from some lexical databases like WordNet [4]. However, such a generic semantic similarity calculating algorithm is not perfectly applicable in domain-based concept systems. Furthermore, Li's work did not handle cases of crossing-layer mappings, which is common in tree mapping where similar terms may be placed in various layers of the trees.

Summarizing, to the best of our knowledge, no research has been fully done to measure similarity of trees based on both structure comparing and concept comparing.

3. Tree Definition for Concept Structure

Tree comparing has been studied in many researches. These researches are mainly focused on finding matches based on pure structure or geometry perspective (e.g. [7, 8]) without considering the conceptual semantics of the tree nodes in a knowledge context.

We extend traditional definition of trees for the sake of modeling concept structures. The formal definition is given below:

Definition 1: Concept Tree. An (unordered labeled) Concept Tree is a six-tuple $T = (V, E, \mathtt{L}^V, \text{root}(T), \mathtt{D}, \mathtt{M})$ where V is a finite set of nodes, E is a set of edges satisfying that $E \subset V \times V$ which implies an irreflexive and antisymmetric relationship between nodes, $\mathtt{L}^V$ is a set of lexicons (terms) for concepts used as node labels, $\text{root}(T) \in V$ is the root of the tree, $\mathtt{D}$ is the domain of discourse, and $\mathtt{M}$ is an injective mapping from V to $\mathtt{L}^V$, $\mathtt{M}: V \rightarrow \mathtt{L}^V$ ensuring that each node has a unique label. For convenience, we simply call each term in $\mathtt{L}^V$ a concept with an agreement of their semantics.

A concept tree is acyclic and directed. If $(u, v) \in E$, we call u a parent of v and v a child of u, denoted as $u = \text{parent}(v)$ or $v = \text{child}(u)$. The set of all children of node u is denoted as $C(u)$. For two nodes $u_1, u_2 \in V$, if $(u_1, u_2) \in E^*$ holds, then we call u_1 an ancestor of u_2 and u_2 a descendant of u_1. The set of all descendants of node u is named $D(u)$.

Definition 2: Conceptual Similarity Measure. A conceptual similarity measure $S_{L^{V1}, L^{V2}}$ is a set of mapping from two lexicon sets $\mathtt{L}^{V1}$, $\mathtt{L}^{V2}$ used in different concept trees to the set of real numbers R, $S_{L^{V1}, L^{V2}}$: $\mathtt{L}^{V1} \times \mathtt{L}^{V2} \rightarrow R$, in which each mapping denotes the conceptual similarity between two concepts represented by these two lexicons. R has a range of (0, 1]. $S_{L^{V1}, L^{V2}}$ is semantically reflexive and symmetric, i.e. for $l_1 \in \mathtt{L}^{V1}$ and $l_2 \in \mathtt{L}^{V2}$ we have $S_{L^{V1}, L^{V2}}(l_1, l_1) = 1$ and $S_{L^{V1}, L^{V2}}(l_1, l_2) = S_{L^{V1}, L^{V2}}(l_2, l_1)$. For convenience, we simply use $w = s(l_1, l_2)$ to refer to the number value of conceptual similarity between two concepts from two trees T_1 and T_2. Intuitively, the larger w is, the closer the two concepts are and $w = 1$ means two concepts are actually synonymous.

Conceptual similarity between two concepts can be given by domain experts or calculated based on some linguistic analysis methods. For instance, Mitra et al use a linguistic matcher to assign a similarity score to a pair of similar concepts [11]. For example, given the strings "*Department of Defense*" and "*Defense Ministry*", the match function returns match(*Defense*, *Defense*) = 1.0 and match(*Department*, *Ministry*) = 0.4, then it calculates the similarity between the two strings are: s("*Department of Defense*", "*Defense Ministry*") = (1 + 0.4)/2 = 0.7.

For $l_1 \in \mathtt{L}^{V1}$ and $l_2 \in \mathtt{L}^{V2}$, if there is no definition for l_1 and l_2 in the measure, we view l_1 and l_2 as totally different concepts. Such a concept pair will not be considered when two concept trees are being compared.

4. Tree Transformation Operations and Transformation Cost

Tree transformation operations can map one tree T into another, T', as are defined below.

4.1. Deleting node *v* (denoted as delete(*v*))

If $v \neq \text{root}(T)$, then $V' = V - \{v\}$, $E' = E - \{(u, v) \mid u = \text{parent}(v)\} - \{(v, v_c) \mid v_c \in C(v)\} + \{(u, v_c) \mid u = \text{parent}(v) \wedge v_c \in C(v)\}$, $\mathtt{L}^{V'} = \mathtt{L}^{V} - \{\mathtt{M}(v)\}$, and $\mathtt{M}' = \mathtt{M} - \{(v, \mathtt{M}(v))\}$.

It must be noted that when deleting one node, besides eliminating that node from the tree, we still need to make its children nodes new direct children nodes of its parent node, which is different from deleting a sub-tree.

If $v = \text{root}(T)$, the result of deleting is a forest $\{T[v_c] \mid v_c \in \mathrm{C}(v)\}$. In a concept tree the root is usually a very general concept like "object", therefore we assume that all trees have a common root concept and restrict that the root is never allowed to be deleted.

4.2. Inserting node *v* under node *u* (denoted as insert$_u$(*v*))

We have

$V' = V + \{v\}$, $E' = E + \{(u, v)\} + \{(v, u_c) \mid u_c \in C'(u)\} - \{(u, u_c) \mid u_c \in C'(u)\}$, $\mathtt{L}^{V'} = \mathtt{L}^{V} + \{l_v\}$, and $\mathtt{M}' = \mathtt{M} + \{(v, l_v)\}$, where l_v is the lexicon assigned to the new node v, and $C'(u) \subseteq C(u)$ meaning that some children nodes of u are changed to be children of the new node v. The elements contained in $C'(u)$ is determined by the context when performing the editing operation.

4.3. Re-labeling node *v* (denoted as relabel$_{l_v \to l_v'}$(*v*))

This is a particular operation in labeled tree. Re-labeling of v with label l_v is to assign v a new label l_v', keeping positions of all nodes unchanged. We have $\mathtt{L}^{V'} = \mathtt{L}^{V} - \{l_v\} + \{l_v'\}$ and $\mathtt{M}' = \mathtt{M} - \{(v, l_v)\} + \{(v, l_v')\}$.

4.4. Moving node *v* to be under node *u* (denoted as move$_u$(*v*))

This is an extended operation in knowledge context that is not defined in classical tree editing operation sets. From Figure 2 we see that in the case of a pure structured tree (a) and (b) two operations delete(E) and insert$_B$(E) can be performed to convert (a) to (b). However, when mapping a concept tree to another we cannot simply delete a node and then insert it since the concept represented by the node's label already exists in the tree.

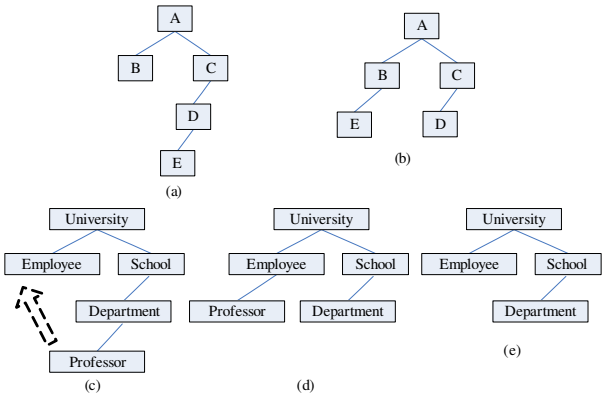

Figure 2. A case for moving operation.

More specifically, in Figure 2 two trees (c) and (d) put the concept "*Professor*" in different positions and by moving node "*Professor*" to be under "*Employee*" we transform (c) to (d), instead of deleting "*Professor*" and then inserting it back (from (c) to (e) then (e) to (d)).

The moving operation regulates that $E' = E + \{(u, v)\} + \{(v, u_c) \mid u_c \in C'(u)\} + \{(\text{parent}(v), v_c) \mid v_c \in C(v)\} - \{(\text{parent}(v), v)\} - \{(v, v_c) \mid v_c \in C(v)\} - \{(u, u_c) \mid u_c \in C'(u)\}$, where $C'(u) \subseteq C(u)$ meaning that some children of node u are changed to be children of the node v based on the operation context.

Definition 3. Transformation Cost. Each transformation operation Op on tree T is mapped to a real number which is defined as the transformation cost of the operation and denoted as $\gamma(Op)$. The transformation cost reflects the extent of change it makes to the tree.

If $OP = \{Op_1, Op_2, \ldots, Op_k\}$ is an transformation sequence, then the transformation cost of the sequence is defined as $\gamma(OP) = \sum_{i=1}^{i=|OP|} \gamma(Op_i)$.

Definition 4. Tree Transformation Cost and Similarity Index. If OP is a transformation sequence mapping a tree T_1 to another tree T_2, then the tree transformation cost of T_1 and T_2 is defined as

$\gamma(T_1 \to T_2) = \min\{\gamma(OP) \mid OP$ is a transformation sequence mapping T_1 to $T_2\}$.

Also, we define similarity index of two trees T_1 and T_2 as

$\gamma(T_1, T_2) = \min\{\gamma(T_1 \to T_2), \gamma(T_2 \to T_1)\}$.

It is a measure representing the degree to which two trees are similar to each other. The higher the tree transformation cost and similarity index is, the less similar the two trees are and vice versa.

5. Computing of Transformation Cost

In a tree transforming process we need to count the total cost of all transformation operations. A tree transforming process that maps tree T_1 into T_2 based on $S_{L^{V_1}, L^{V_2}}$ contains the following tasks:

(1) Compute the set of nodes to be deleted, D, in T_1.

$D = \{u \mid u \in V_1 \wedge \mathtt{M}_1(u) \notin \mathtt{L}^{V2} \wedge \neg\exists s(\mathtt{M}_1(u), l_2) \in S_{L^{V_1}, L^{V_2}} (l_2 \in \mathtt{L}^{V2})\}$. That is, the nodes which labels are appearing in T_1 but T_2 and have no conceptual similarity with any labels in T_2 defined.

(2) Compute the set of nodes to be inserted into T_1, I.

$I = \{v \mid v \in V_2 \wedge \mathtt{M}_2(v) \notin \mathtt{L}^{V1} \wedge \neg\exists s(l_1, \mathtt{M}_2(v)) \in S_{L^{V_1}, L^{V_2}} (l_1 \in \mathtt{L}^{V1})\}$. That is, the nodes which labels are appearing in T_2 but T_1 and do not have conceptual similarity definition with any labels in T_1.

(3) Try every possible combination of the deletion and insertion operations and find the minimal cost.

(4) Compute the set of nodes to be moved within T_1 itself, M, and move them.

$M = \{u \mid u \in V_1 \wedge (\mathtt{M}_1(u) \in \mathtt{L}^{V2} \wedge \mathtt{M}_1(\text{parent}(u)) \neq \mathtt{M}_2(\text{parent}(\mathtt{M}_2^{-1}(\mathtt{M}_1(u)))) \wedge \neg\exists s(\mathtt{M}_1(\text{parent}(u)), \mathtt{M}_2(\text{parent}(\mathtt{M}_2^{-1}(\mathtt{M}_1(u))))) \in S_{L^{V_1}, L^{V_2}}) \vee (\exists s(\mathtt{M}_1(u), l_2) \in S_{L^{V_1}, L^{V_2}} (l_2 \in \mathtt{L}^{V2})) \wedge \mathtt{M}_1(\text{parent}(u)) \neq \mathtt{M}_2(\text{parent}(\mathtt{M}_2^{-1}(l_2))) \wedge \neg\exists s(\mathtt{M}_1(\text{parent}(u)), \mathtt{M}_2(\text{parent}(\mathtt{M}_2^{-1}(l_2)))) \in S_{L^{V_1}, L^{V_2}})\}$. That is, the nodes that are appearing in both T_1 and also in T_2, or which labels have conceptual similarity with labels defined in T_2, but which parents are neither the same nor similar.

(5) After the deleting, inserting, and moving operations performed on T_1, T_1 now has the same structure with T_2, but still has some nodes with different labels (implying different conceptual semantics). The final task is to compute the set of nodes to be re-labeled, R, and re-label them. $R = \{u \mid u \in V_1 \wedge \mathtt{M}_1(u) \notin \mathtt{L}^{V2} \wedge \exists s(\mathtt{M}_1(u), l_2) \in S_{L^{V_1}, L^{V_2}} (l_2 \in \mathtt{L}^{V2})\}$. That is, the nodes that are appearing in both T_1 and T_2 with different labels, but the labels have conceptual similarity between them.

Let OP be the editing sequence containing operations in the above tasks, the transforming cost is computed as follows (using pure operation names):

$$\gamma_{T1 \to T2}(OP) = \min\{\sum_{i \in D} \gamma(delete(i)) + \sum_{i \in I} \gamma(insert(i)) + \sum_{i \in M} \gamma(move(i)) + \sum_{i \in R} \gamma(relabel(i))\}$$

The cost of each transformation operation is a key issue for the measuring. The cost is affected by the level that a node resides in the tree structure, the scale of the set of nodes, the number of descendants of a node, and the similarity of two concepts attached to two nodes. For example, first, a node at a higher layer contains richer semantics than a lower node does, or, the concept it represents is more significant for the domain than a lower one does. Therefore, when a node u is at a higher layer, the effect to the concept tree of deleting u or inserting a new node under u is larger than that of deleting or inserting a node at a lower layer. Second, the more nodes a tree has, the less the effect will be when one node is deleted or inserted. That is, the larger the concept tree is, the less different it will be if it gets one new concept or loses one old concept. Third, a node with more descendants will cause greater change to the tree structure if it is deleted, or greater change if a node gets more descendants after it is inserted. Finally, the more similar the two concepts are, the less the cost will be to change one into another one.

According to the research of [1, 9] and our own observations, we introduce the following cost computing algorithms:

- Deleting cost.

$$\gamma(delete(v)) = \frac{height(T) - depth(v) + 1 + |D(v)|}{|V|},$$

where v is a non-root node, height(T) is a function calculating the height of tree T, depth(v) calculates the depth of node v, and $|D(v)|$ is the number of descendants of node v (including its direct children and indirect offspring). Intuitively, depth(root(T)) = 1, and depth(v) > 1 iff v is not the root. If v is a leaf node, $D(v) = \varnothing$ and $|D(v)| = 0$. When v is a leaf node at the lowest level (height(T) = depth(v)), deleting v will cause the minimal effect to the tree and $\gamma(\text{delete}(v)) = 1/|V|$. Note that here V refers to the original node set before the deletion.

- Inserting cost.

$$\gamma(insert_u(v)) = \frac{height(T) - depth(u) + 1 + |D(v)|}{|V|},$$

where $|D(v)|$ is the number of descendants that v gets after it is inserted. Note that here V refers to the original node set before the insertion. When u is at the lowest layer, inserting a new node v under u will result in the minimal cost $\gamma(\text{insert}_u(v)) = 1/|V|$.

- Moving cost.

$$\gamma(move_u(v)) = \frac{1}{2}[\gamma(delete(v)) + \gamma(insert_u(v))] \times \frac{|V| - 2}{|V|}$$

, where $|V|>2$ (the tree has a root and at least two non-root node) and $u \neq \text{parent}(v)$. Note that here $\text{insert}_u(v)$ is performed on a tree without node v. In this definition we consider both deleting and inserting operations because the moving operation does generate effects similar to deleting and inserting, although not exactly the same. The factor 1/2 adjusts the cost of operations since the node is not truly deleted and inserted into the

tree. Another factor (|*V*| - 2)/|*V*| adjusts the cost again to ensure that in an extreme case where *v* is the only node other than the root, its moving cost should be 0 (actually it cannot be moved) and when the number of nodes in the tree grows, the effect of the moving operation to the tree structure turns weaker.

- Re-labeling cost.

This cost is heavily dependent on the similarity of two labels (concepts). Re-labeling cost is different from deleting cost, inserting cost, or moving cost since the re-labeling operation does not result in change of a tree structure. The cost is heavily dependent on the similarity of two labels (concepts). Kouylekov et al [12] proposed a definition for substitution of two similar words w_1, w_2 as $\gamma(\text{insert}(w_2))\times(1 - \text{sim}(w_1, w_2))$ where insert(w_2) is the cost of inserting w_2 and sim(w_1, w_2) is the similarity between w_1 and w_2. This definition does not take the deletion of the original word into consideration, therefore when two words have no conceptual similarity the cost of substitution becomes the cost of insertion, neglecting the implicit deleting operation. In our work we give a more comprehensive definition.

Let the similarity measure between two concepts l_{v1}, l_{v2} which are attached to node *v* be *s*, $0 \le s \le 1$, we define:

$$\gamma(relabel_{l_{v1}\to l_{v2}}(v)) = [\gamma(delete(v)) + \gamma(insert_{parent(v)}(v))]\times(1-s)$$

We analyze two extreme cases: if $s = 1$, then re-labeling will only result in literal replacing without any loss of information, therefore the re-labeling cost is 0; if $s = 0$ (i.e., the two concepts are totally different), the re-labeling operation is equivalent to deleting *v* and inserting *v* again, the transformation cost is $\gamma(\text{delete}(v)) + \gamma(\text{insert}_{\text{parent}(v)}(v))$. In other cases, the cost will be between these two boundaries.

6. Application on Ontology Comparison

In the situations where two trees should be compared not only based on their geometrical structures but also concept hierarchy implicated by their structures, our methodology can be applied to measure their similarity in a knowledge context. The integration of ontologies is among such situations. In ontology integration, we consider the following two tasks that require ontology comparison to be crucial:

(1) Before starting the integration, find from the original candidate ontologies one that is much more similar to most of others (meaning that it is possibly a better one) and take it as a foundation to initialize the integration.

(2) Or, after the integration process is finished, compare the global ontology obtained with the original candidate ontologies to find the best one among them that is the most similar to the integrated result, i.e., to evaluate the trustability of original candidate ontologies.

Both the above two tasks require some way to measure the similarity of different ontologies (composed by concepts and relationships, therefore bearing structural characteristic), other than just the similarity of two individual concepts.

In many cases an ontology can be organized into a tree structure where each node presents one concept, semantics of relationships between concepts are identical (e.g. "part-of"), and each concept is related to only parent concept [10]. Our methodology is able to help evaluate similarity among different ontologies.

Figure 1 shows three simplified ontologies for the university domain. Given that a set of measures describing the similarity of some concept pairs is defined:

- *s*(*People*, *Human*) = 1;
- *s*(*Registered Student*, *Student*) = 1, and
- *s*(*Faculty*, *Professor*) = 0.9.

One transformation sequence mapping T_1 to T_2 causes the following costs:

(1) $\gamma(\text{delete}(\textit{Student Residence})) = 2/7 = 0.29$;
(2) $\gamma(\text{insert}_{\textit{University}}(\textit{Organization})) = 4/6 = 0.67$ (making *Department* a child node of *Organization*);
(3) $\gamma(\text{insert}_{\textit{Organization}}(\textit{Library})) = 2/7 = 0.29$;
(4) $\gamma(\text{move}_{\textit{Organization}}(\textit{Research Center})) = (1/2)((1/4) + (2/7))(6/8) = 0.20$;
(5) $\gamma(\text{relabel}_{\textit{People}\to\textit{Human}}(\textit{People})) = 0$;
(6) $\gamma(\text{relabel}_{\textit{Registered Student}\to\textit{Student}}(\textit{Registered Student})) = 0$;
(7) $\gamma(\text{relabel}_{\textit{Faculty}\to\textit{Professor}}(\textit{Faculty})) = 0.041$.

Finally, the entire tree transformation cost is 1.491. We have to point out that compared with the deleting, inserting, and moving costs, the re-labeling operation has a minimal effect on the tree, therefore its cost is much smaller than the cost of the other three types.

Usually there are various ways to map one tree into another with different costs. For instance, in the university case, if both "*Department*" and "*Research Center*" are made child nodes of "*Organization*" when inserting "*Organization*", the inserting cost will change to 0.83, and there will be no moving cost, therefore the entire matching cost becomes 1.451.

The following Table 1 summarizes the transformation cost and similarity index between three trees:

Table 1. Transformation cost and similarity index.

Tree Pair	Transformation Cost		Similarity Index
T_1, T_2	$\gamma(T_1\to T_2)$	$\gamma(T_2\to T_1)$	$\gamma(T_1, T_2)$
	1.227	1.034	1.034
T_1, T_3	$\gamma(T_1\to T_3)$	$\gamma(T_3\to T_1)$	$\gamma(T_1, T_3)$
	2.839	2.614	2.614
T_2, T_3	$\gamma(T_2\to T_3)$	$\gamma(T_3\to T_2)$	$\gamma(T_2, T_3)$
	2.128	2.039	2.039

Since T_2 is closer to both T_1 and T_3, it is better to be used as a foundation for integration.

On the other hand, if T_2 is the result of integration based on T_1 and T_3, T_1 can be claimed more trustable since it is closer to the common ontology (T_2) in terms of both structure and knowledge contained in its structure.

7. Conclusion and Future Work

In this work we extend classical similarity measuring methodology based on tree editing operations to make it more applicable to trees that are modeling concept structures. We propose definitions on tree transformation operations and transformation costs, based on how the similarity of two concept structures can be measured. We apply this methodology to ontology comparison where different ontologies of one domain can be represented as trees and relationships between concepts are identical. By discovering similarity between ontologies we are able to choose the best one from all the candidates and make it the foundation of the integration. Also the trustability of these ontologies can be evaluated.

In our next step we will extend the tree structure to a graph which can model more complex concepts and relationships. New definitions for graph transformation operation and transformation cost are to be explored. Meanwhile, more types of relationships among concepts have to be considered, which requires further considerations on the semantics of the relationships.

8. References

[1] W. Shen, D. H. Norrie and J. A. Barthes, "*Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing"*, Taylor & Francis, 2001.

[2] M. Guegan and N. Hernandez, "Recognizing Textual Parallelisms with Edit Distance and Similarity Degree", *Proceedings of EACL 2006*, The Association for Computer Linguistics, Trento, Italy, April 3-7, 2006.

[3] J. Allali and M. Sagot, "Novel Tree Edit Operations for RNA Secondary Structure Comparison", *Proceedings of IWABI 2004*, Bergen, Norway, September 17-21, 2004, pp. 412-425.

[4] M. Warin, H. Oxhammark and M. Volk, "Enriching An Ontology with WordNet based on Similarity Measures", *Proceedings of the MEANING-2005 Workshop*, Trento, Italy, February, 2005.

[5] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", *The Morgan Kaufmann Series in Data Management Systems*, Jim Gray, Series Editor. Morgan Kaufmann Publishers, August 2000.

[6] S. Pinto, A. Gómez-Pérez and J. P. Martins, "Some Issues on Ontology Integration". *Proceedings of IJCAI1999 Workshop on Ontologies and Problem Solving Methods*, 1999.

[7] S. Guda, H. V. Jagadish, N. Koudas, D. Srivastava and T. Yu, "Approximate XML Joins", *Proceedings of ACM SIGMID*, 2002.

[8] J. Jin, B. K. Sarker, V. C. Bhavsar, H. Boley and L. Yang, "Towards a Weighted-Tree Similarity Algorithm for RNA Secondary Structure Comparison", *Proceedings of HPC Asia 2005*, IEEE Computer Society, pp. 639-644, 2005.

[9] N. Guarino, "Understanding, Building and Using Ontologies", *International Journal of Human-Computer Studies*, 46(2/3) (1997), pp. 293-310.

[10] M. Cho, H. Kim and P. Kim, "A New Method for Ontology Merging based on Concept using WordNet", *Proceedings of ICACT 2006*, Volume 3, pp. 1573-1576, 2006.

[11] P. Mitra and G. Wiederhold, "Resolving Terminological Heterogeneity in Ontologies", *Proceedings of the ECAI'02 Workshop on Ontologies and Semantic Interoperability*, 2002.

[12] M. Kouylekov and B. Magnini, "Recognizing Textual Entailment with Tree Edit Distance Algorithms", *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, UK, 2005.

[13] J. T. Yao and M. Zhang, "A Fast Tree Pattern Matching Algorithm for XML Query", *Proceedings of International Conference on Web Intelligence (WI 2004)*, 2004, pp. 235-241.

[14] N. Bruno, D. Srivastava and N. Koudas, "Holistic Twig Joins: Optimal XML Pattern Matching", *Proceedings of SIGMOD*, 2002, pp. 310-321.

[15] A. V. Aho, M. Ganapathi and S. W. K. Tjiang, "Code Generation Using Tree Matching and Dynamic Programming", *ACM Transactions on Programming Languages and Systems*, Vol. 11, Issue 4 (October 1989), pp. 491-516.

[16] A. Maedche and S. Staab, "Measuring Similarity between Ontologies", *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, *Ontologies and the Semantic Web*. Lecture Notes in Computer Science, Vol. 2473 (2002), pp. 251-263.

[17] S. Li, H. Hu and X. Hu, "An Ontology Mapping Method Based on Tree Structure", *Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKD'06)*, Guilin, Guangxi, China, 2006, pp. 87-88.