PTree: A Tool to Draw Tree for Concept Relation Tree (CRT)

Mohd. Hasan Selamat[2], Wan Malini Wan Isa[2], Jamaliah Abdul Hamid[1], Hamidah Ibrahim[2], Rusli Abdullah[2] and Nurul Amelina Nasharuddin[2]

[1]Faculty of Educational Studies
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor
Tel: 03-89468177, Fax: 03-89468246
E-mail: aliah@putra.edu.my

[2]Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor
Tel: 03-89466555, Fax: 03-89466576
E-mail: hasan@fsktm.upm.edu.my, wanmalini84@gmail.com, hamidah@fsktm.upm.edu.my, rusli@fsktm.upm.edu.my, nurulamelina@gmail.com

ABSTRACT

Our research project is currently to develop an Automatic Meaning Extraction (AME) System which automatically extracts concepts and their relationships across texts in all domains of knowledge. Concept Relational Tree (CRT) is one of the text analyzer applications used in the AME System to automatically extract concepts and their relationships in a document. To check on the correctness of the extraction of concepts and their relationships, the PTree is designed to reconstruct the text by reverse input. In this paper we present the PTree tool to test the accuracy of the automatic tagging and tree structure created by CRT from texts. The PTree tool is implemented from Java Universal Network/ Graph Framework (JUNG) libraries. This tool provides a few functions to allow for flexibility in drawing relational trees for concepts. Due to its flexibility and dynamic features, PTree can be further extended for use in the deconstruction of highly complex texts.

Keywords

Parse tree, Java Universal Network/Graph, interface

1.0 INTRODUCTION

The AME System is an ongoing development of a system that extracts concepts and their relationships automatically across domains of knowledge. AME when fully developed is an automatic knowledge extraction system for the building of knowledge ontologies and it will hopefully allow the extraction and integration of ontologies of various domains to enable comprehensive mapping of knowledge. At the present stage of development, AME is able to visualize concepts and their relationship from a collection of documents and also enables the trace-back of particular knowledge schema back to its original source in a document.

In the process to extract concepts from text and their relational mapping, the AME system uses a few text analyzer applications which are Concept Relational Tree (CRT), Connector Based Extraction (CBE), Concept Relational Parser (CRP), and Social Competition Model (SCM). These innovative applications are used to enable automatic and more refined extraction of concepts and concepts relationships to finally generate semantic schemas. When AME deconstruct text, it performs tagging based on concepts, relations and attributes. The CRT plays a crucial role in arranging these concepts, relations and attributes so that the semantic hierarchy is maintained even as texts get more complex. CRT is an application which enhances the architecture of Discourse Structure Tree (DST) by integrating it with another tree, called the expression tree (ET) (See Figure 1, 2 and 3). DST organizes semantics hierarchically through markers. Since discourse markers are not widely used in text, the applicability of DST is compromised. ET on the other hand, improves coverage and granularity by providing a new framework for semantic organization based on connectors rather than discourse markers.

The integration of ET and DST provides grater control to the interpretation of semantics since semantics can be attained at various levels of tree. Conventionally, lengthy and complex texts cause the tree to increase in size horizontally, thus endangering loss of the connectivity from original agent. CRT maintains the relational connectivity by vertical expression. CRT improves semantic organization for most type of text. The more refined the semantic organization is, the better the approach becomes to semantic deconstruction. This in turn will enable better schemes of meaning to be extracted and linked to one another. By improving cohesion of schemes, the model consequently will enhance the accuracy of the text understanding.

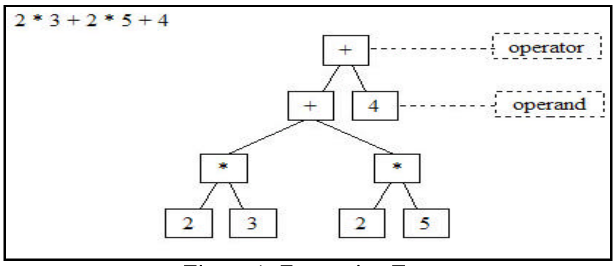
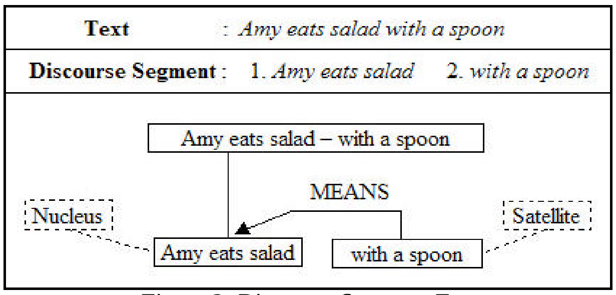


Figure 1: Expression Tree



Text : Amy eats salad with a spoon
Discourse Segment : 1. Amy eats salad 2. with a spoon
Amy eats salad – with a spoon
MEANS
Nucleus
Satellite
Amy eats salad
with a spoon

Figure 2: Discourse Structure Tree

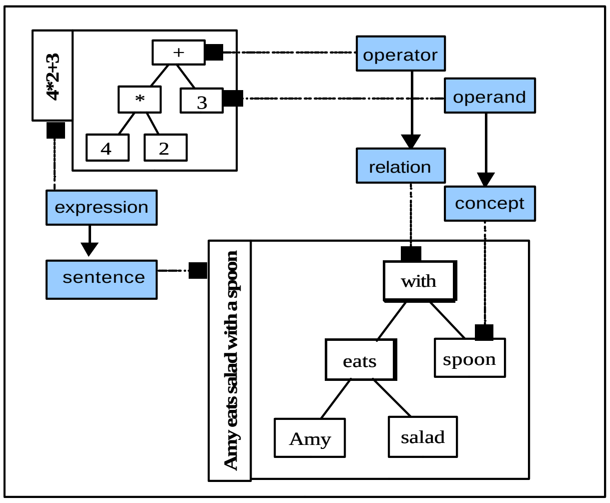
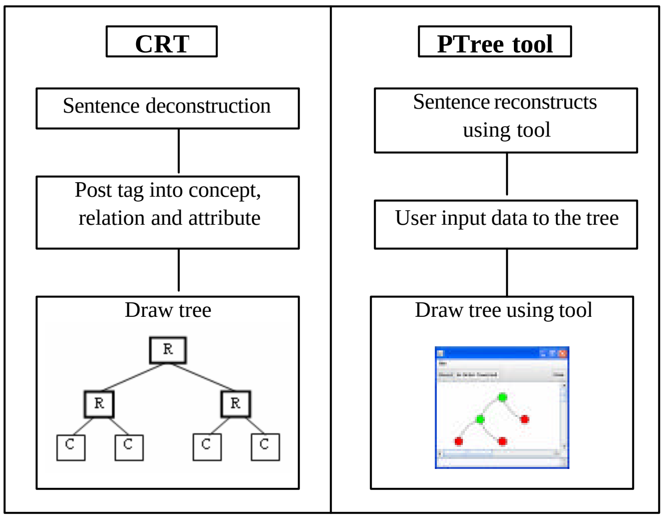


Figure 3: Expression Tree and CRT

Experimentation is required to show the accuracy of CRT in performing automatic sentences deconstruction. PTree enables those concepts and relations as identified by the CRT to be re-inserted by the user at various levels of parent and child nodes, and PTree then reconstructs the entire sentence. We can then compare the sentence reconstructed by PTree based on the user insertions to the sentence which had been automatically deconstructed by CRT. Figure 4 show the flow of sentence deconstruction by CRT and sentence reconstruction by PTree tool. This tool is developed using JAVA language and other software library is called Java Universal Network/Graph Framework (JUNG).



CRT
Sentence deconstruction
Post tag into concept, relation and attribute
Draw tree
R
R
R
C
C
C
C
PTree tool
Sentence reconstructs using tool
User input data to the tree
Draw tree using tool

Figure 4: Sentence deconstruction and reconstruction

2.0 RELATED WORK

A tree is composed of a collection of nodes, where each node has some associated data and a set of children. A node's children are those nodes that appear immediately beneath the node itself. A node's parent is the node immediately above it. A node which has no parent is called root node (Mitchell, 2008). There are several trees commonly used in computer science and natural language processing. Binary tree is one used in computer science field. A binary tree is a data structure tree in which each node has at most two children. Each child of a node is designated to its left or right. Nodes that have no children are referred to as leaf nodes while nodes that have one or two children are referred to as internal nodes. Examp les of binary tree are complete binary tree, full binary tree, binary search tree, binary heap, balanced binary tree and many more (Black, 2007). In binary tree, the nodes only can contain numbers as the content or label of the nodes.

Parse tree on the other hand is commonly generated for sentences in natural languages, as well as during processing of computer languages, such as programming languages. A parse tree is a tree that arranges the words in the sentence according to their part-of-speech tag and production rules. The production rules determine the hierarchical manner of which tags are related to one another by specifying the formula of tag decomposition. Consider an example of a parse tree (See Figure 5). The leaves of the tree consist of words fro m the sentence (Ungku Chulan, 2007).

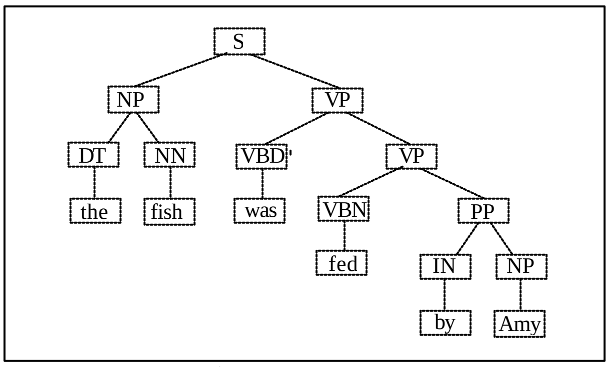


Figure 5: Parse Tree.

From the concept of binary and parse tree, we develop a tool which is combination from the two concepts. PTree is a tool like binary tree in that one parent has two nodes and the children are in left and right positions. Because of the limitations of labeling the content in binary tree, PTree uses the concept of parse tree to enable the use of words, not numbers, to label the content in this tool. PTree tool were designed synonymously for CRT where the node in the tree contain concept (C) and relation (R) node. The basic C-R-C tree structure in CRT is itself innovative, doing away with the need to tag determiners and conjunctions, adjectives and adverbs. The classification of the new part of speech tagging in the AME system is described in another paper.

JUNG is a software library that provides a common and extendible language for the modeling, analysis and visualization of data to enable it to be represented as a graph or network (O'Madadhain, Fisher, Smyth, White & Boey, 2003). JUNG library can be implemented in any Java-based applications and makes use of the extensive capabilities of the Java API. One of the functionalities in JUNG is that it is suitable for creating trees, that is, it provides a mechanism for annotating metadata to the graphs, entities and relations. This facilitates the creation of analytic tools for complex data sets to enable one to examine the relations between entities as well as the metadata attached to each entity and relation.

3.0 PTree TOOL

3.1 Structure of PTree tool

PTree is a tool to reconstruct sentences from a tree. The tree uses the same concept as a binary tree whereby one parent has two children. This tree is made of a node, a combination of nodes or combination of trees. The node of the tree is made of either concept or relation. The internal nodes are all relations. They can be seen as the branches of the tree and the external nodes are all concepts. They are the leaves of the tree. Relation nodes describe the connection of nodes in the tree or between two trees under it. PTree tool is different from normal parse tree whereby it allows the node in the tree to be either concepts or relations, and not the usual part of speech tags.

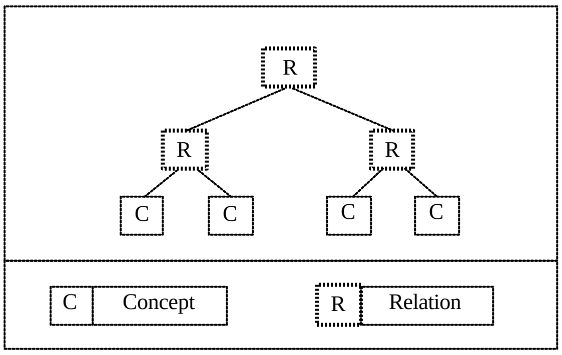


Figure 6: Concept Relation Tree Structure

3.2 Functions in PTree

PTree tool contains menu bar, tool bar, tree viewer and text area. When the user runs this tool, one node will appear at the center of the tree viewer which is called the root node. From a single root node, user can extend the node to become a tree. To build a tree, user right clicks on the node and a pop up menu will display. The pop up menu consists of several functions that user can use to interactively draw the tree.

First function is "**Add node (L&R)**" where this function is to add left and right nodes to an existing node. Like the binary tree, the parent node only can have two children. If the parent node already has two children, the user cannot perform this function to add left or right nodes again and an error message will appear on the screen. However, the PTree now allows for either the left child node or the right child node to be extended further, but not both. User can do this by choosing "**Add node (L)**" or "**Add node (R)**". When the next level of child nodes have been extended, the left or right of the previous level now changes its type from concept to relation.

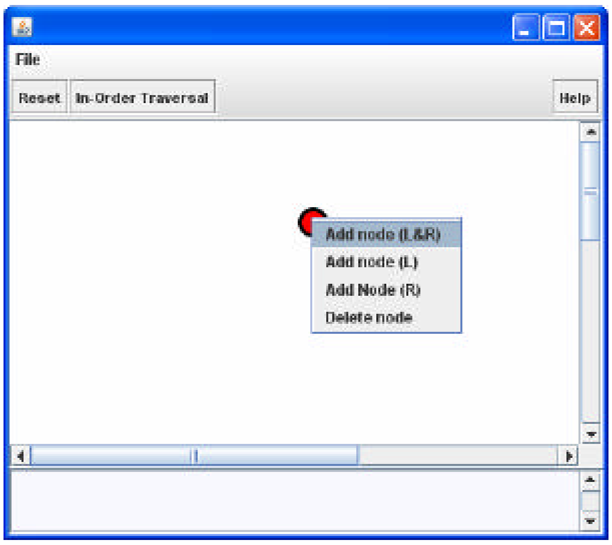


Figure 7: Pop up menu function.

For example in Figure 8 (left) node 1 is a concept but when user adds the child node to node 1, it changes its type from concept to relation. See Figure 8 (right). This allow for new levels of concepts to be introduced and related to one another. In this way, the parent node maintains the initial pivotal relation that encapsulates all other relationships spawned by n-levels of nodes. It is this ability of PTree to maintain the pivotal relationship at the parent node that enables the tree to hold on the essential meaning of a sentence in spite of any number of subsidiary concepts and their relationships are added on.

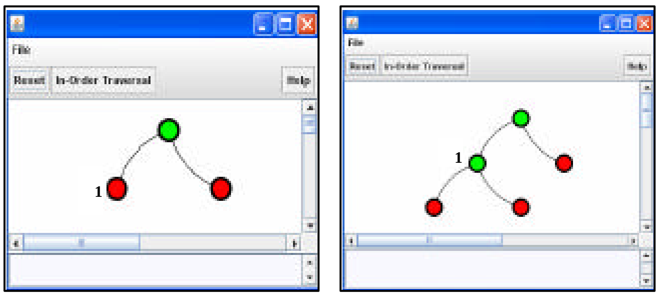


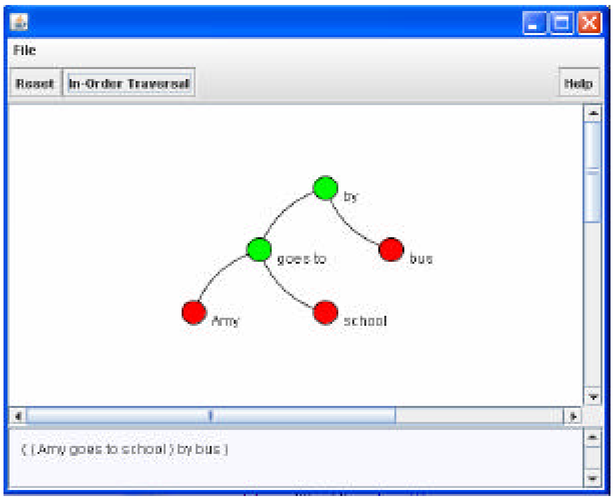
Figure 8: Type node change from concept to relation

Thus, a sentence “Amy goes to the school by bus which is driven by Ali on Mondays of every month” will never lose its essential core of meaning that “Amy goes to the school by bus” which happens to be modified by “bus is driven by Ali”, “Ali drives on Mondays” and “Mondays of every month” in the layer of child nodes. The parent node is always the connector that links the concepts in the left or right children nodes.

Another function on pop up menu is “**Delete node**”. When the user performs this function on a child node, the node will be deleted. If the user performs this function on a parent node, all the child nodes belong to the parent will be deleted including the parent node.

This PTree tool enables the user to input (insert) the content to the node. User can perform this action by double clicking at the node and a pop up input dialog will be displayed. User enters the content for the node and it is then displayed as a label below the node. This makes it easy for the user to insert and view the content of the node. If the user wants to change the content of the node, the user double clicks on the node and enters the new content in input dialog box. The content of the node are not limited only to numbers or words. This input function gives the flexibility to change the status of the node to either concept or relation. In conventional binary trees, the rigid notation by numbers preserves the order of the tree hierarchy, and difficulty arises when sentences become more complex, with many subsidiary concepts.

The tool bar in this PTree has three formatting buttons. First button is “**Reset**” button. This button is to reset the node to the earlier position, clear all the nodes and their contents and draw a new root node. The next button is “**In-Order Traversal**” button. When user clicks this button, the tool will traverse the tree from left to right node to compute or read all the nodes of the left subtree, the root and lastly the right subtree. As a result of the traversal, the reconstructed sentence will be displayed in the text area. The reconstructed sentence provides semantic checks to whether CRT has correctly extracted concepts and their relations. If CRT extraction and relation parsing was correct, then the sentence parsed together (sentence reconstruction) by PTree should be the same sentence that was deconstructed by AME. Our experiments with the PTree show at this point that the CRT is capable of 67% accuracy.



((Amy goes to school) by bus)

Figure 9: PTree tool

This tool also has a menu bar which contains ‘**Save as Image**”, “**Print**” and “**Exit**” function. Save as image function allows user to save the tree as image in JPEG file format. Print function will connect this tool to the online printer and allows the user to print the tree directly from this tool. User can click on exit function to exit from this tool.

4.0 CONCLUSIONS AND FUTURE WORK

This PTree tool was developed to draw a tree to test the accuracy of CRT in the AME system. The experiment is important to check whether CRT has correctly extracted the concepts and relations from the sentences in a document. By comparing the reconstructed sentence produced from the PTree to the original sentence parsed automatically by CRT, the accuracy of the CRT can be determined.

Currently PTree only have one type of traversal which is in-order traversal. The tree reads the content of the node from left subtree, root and the right subtree. For next development of PTree tool, we intend to add other types of traversal such as post-order traversal and pre-order traversal. Post-order traversal means traversing from left subtree, right subtree and finally to the root node. On the other hand pre-order traversal traverse from root node, left subtree and right subtree. These traversals become important in next development to make sure this tool can be use in multiple types of natural language texts.

Besides that, the PTree will be developed to allow for functions such as to add new roots for children node, or to cut and paste subtrees at other subtree. This function is to help users to easily alter the tree when the user makes a mistake during the creation of the tree. Using this function, user does not have to delete all nodes to begin all over again when the tree encounters a mistake. "Add new root" function will also be added to enable users to add node at the highest hierarchy of the tree. PTree is hoped to be a useful tool for natural language processing.

REFERENCES

Barthelemy, F.,Boullier, P.,Deschamp, P.,Kaouane, L., Khajour, A., & Clergerie, E.V. (2001). Tool and resources for Tree Adjoining Grammars. In *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*, Volume 15.

Black, P. E. (2007). Binary tree. Retrieved January 7, 2008, from http://www.nist.gov/dads/HTML/binarytree.html

Cohen, R. F., Battista, G. D., Tamassia, R., Tollis, I. G. & Bertolazzi, P. (1992). A framework for Dynamic Graph Drawing. In *Annual Symposium on Computational Geometry, Proceedings of the eighth annual symposium on Computational geometry*.

Gross, J., & Yellen, J. (1999). *Graph Theory and Its Applications*. Boca Raton, Florida: CRC Press.

Hanrahan, P. (2001). To Draw a Tree. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'01)*.

Kennedy , J.. Parse Trees (n.d). Retrieved January 7, 2008 from http://homepage.smc.edu/kennedy_john/PARSETREES.PDF.

Mitchell, S. (2008). Binary Trees and BSTs. Retrieved January 4, 2008, from http://msdn2.microsoft.com/en-us/library/ms379572(VS.80).aspx .

Moen, S. (1990). Drawing Dynamic Trees. In *IEEE Software*, Volume 7, Issue 4, Page 21-28.

O'Madadhain, J., Fisher, D., Smyth, P., White, S., & Boey, YB. (n.d). Analysis and visualization of network data using JUNG. In *Journal of Statistical Software*. Retrieved January 2, 2008, from http://jung.sourceforge.net/doc/JUNG_journal.pdf.

Power, J. F. & Molly B. A. (2002). Program annotation in XML: a parse-tree based approach. In *Proceeding of the Ninth Working Conference on Reverse Engineering (WCRE'02)*.

Ungku Chulan, U.A. (2007). Connector-*based Extraction with Concept Relational Parser for Extracting Semantic Relation from Text*. PhD thesis, Universiti Putra Malaysia.

Zanden, B. V., & Beeler, M. (n.d). A Tool for Sketching and Manipulating Binary Heaps. Retrieved January 3, 2008 from *http://www.cs.utk.edu/~bvz/HeapAnimation.pdf*