MyTree: A Convenient Toolkit of Parsing Tree Converter

Li Ziyao, 1500017776

Apr. 28th, 2018

Introduction

MyTree is a small python toolkit converting a phrase parsing tree into a dependency parsing tree. Manually designed rules made according to expert knowledge are used to perform the converting process.

MyTree is operated corresponding to nltk.tree. Cooperate MyTree with nltk.tree module, can a user easily transform a bracket-style-annotated phrase parsing tree into a dependency parsing tree, and output it in a ConLL style.

API

`MyTree`, class

Methods

`__init__(self, tree, parent_node=None, current_layer=0)`

Construct a MyTree object corresponding to an input tree.

tree: an object of class `Tree` in `nltk.tree`, or of a self-defined class supporting method `tree.label()` and `tree.__getitem__()`. Note that if is the latter situation, the leaf nodes must be plain text, i.e. the original words of the parsed sentence.

_parent_node: please don't provide this argument.

_current_layer: please don't provide this argument.

Return: constructed MyTree object.

`__geiitem__(self, i)`

Return the i-th child (sub-tree) of the tree's root.

i: index, int.

Return: a sub-tree with the i-th child as its root.

`labelDependencyTree(self, _current_n_words=1)`

Constructing a redundant parent-labeled dependency tree using empirical rules.

_current_n_words: please don't provide this argument.

Return: None

`findDependencyParent(self)`

Cleaning the redundancy of the parent-labeled dependency tree.

Note that this function is only available if `labelDependencyTree()` has been called.

Return: None

`printDependencyConLL(self)`

Return a ConLL-style dependency parsing tree string

Note that this function is only available if `labelDependencyTree()` and `findDependencyParent()` have been called.

Return: string in ConLL-style of the formulated dependency parsing tree.

Attributes:

parent

The parent node of the current tree. `None` indicates that current node is a root of a parsing tree.

layer

The number of layers the current node is on the original parsing tree.

label

Parsing labels for non-terminal nodes and plain text for terminal nodes.

terminal

Whether this node is a terminal node.

children

List of child nodes (or sub-trees) of the current node.

n_children

Number of children of the current node.

id

Only available if `labelDependencyTree()` has been called. Words' id for terminal nodes, and head-words' id for non-terminal nodes.

dependencyParent

Only available for terminal nodes and only if `labelDependencyTree()` and `findDependencyParent()` have been called. The parent node id for current terminal nodes (i.e. the plain word). Used for constructing dependency parsing trees with parent representation.

Notes

Scores of similarities Score for UD: 57.46

Score for SD: 69.28

Score for MELT: 85.99

A coordination adjustment in *Tapping the implicit information for the PS to DS conversion of the Chinese Treebank* is adapted. This adjustment is different than those adopted in all three validation datasets, so the result is slightly different and similarities are therefore lower.

The relevant rules are manually extracted from the MELT dataset, so the score for it is highest among three. This also shows that different understandings of grammars can lead to significant difference in predicting results.