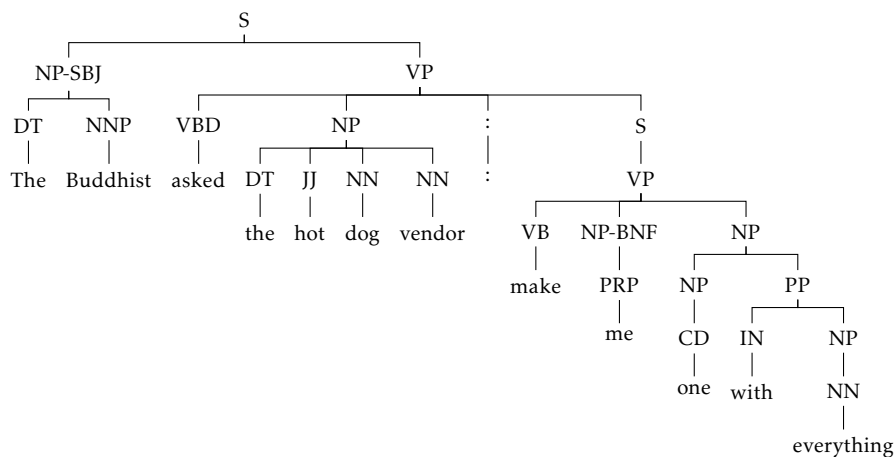


Dependency Parsing exercises:

Dependency Structures Part 2

Deadline: 02.05.2017. You should complete at least three out of five exercises given below. Please send completed solutions to jakub.waszczyk@phil.uni-duesseldorf.de with subject "dependency homework" and an attachment "ex1_lastname(s).pdf".

1. Convert the following constituency analysis into a dependency analysis:



This tree uses Penn treebank annotation: <http://www.surdeanu.info/mihai/teaching/ista555-fall13/readings/PennTreebankConstituents.html>. Not all constituents have a grammatical function; for example, it is implied that an NP under a VP is an object. Treat the BNF (benefactor) label as an indirect object.

- (a) Mark one child as the head in each constituent. Note that in a phrase-structure tree, not only words can be heads, but also phrases. Heuristically, the head of an NP is its rightmost noun or another NP, and the head of S or VP is the main verb or another VP.
 - (b) Add the implied function labels.
 - (c) Now you have enough information to do a conversion. The dependents of a head word consist of its direct siblings and the head words under siblings of ancestors. Convert the tree step-by-step: start with the main verb of the sentence and identify its dependents. Identify dependents of those dependents, etc., until each word has a dependency relation.
 - (d) Is there any information lost in the conversion? Would that matter to a downstream application? Discuss.
2. It is impossible to distinguish between phrase modification and head modification in unlabeled dependency structure (see the lecture). Give an example of a sentence which cannot be properly analyzed because of this property and explain what the issue with the corresponding dependency analysis is. You can assume the UD annotation scheme. Hint: one of the phenomena difficult to handle with dependency structures is coordination.

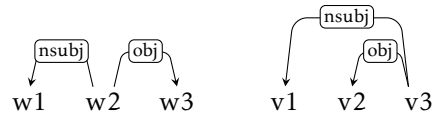


Figure 1: Two structurally identical dependency structures. Node mapping: $\{(w_1, v_1), (w_2, v_2), (w_3, v_3)\}$.

UPDATE (26.04): to show that sentence A cannot be properly analyzed, you can contrast it with another sentence B which (i) syntactically differs from A , and yet (ii) it receives the same UD dependency analysis as A .

Note that dependency structures are, by design, word order-insensitive. Therefore, you can consider two dependency structures (V_1, A_1) ¹ and (V_2, A_2) structurally identical (i.e., representing the same dependency analysis) if there is a one-to-one correspondence (bijection) between V_1 and V_2 which preserves the relations between nodes and their labels.

For instance, the dependency structures depicted in Fig. ?? are structurally identical.

3. In UD, `obj` is used to represent both complements and modifiers. However, distinguishing complements from modifiers is both linguistically and practically motivated.
 - Does UD provide any means of dealing with this issue? Or is it a fundamental flaw, which cannot be resolved without significantly changing the syntactic theory of UD?
 - Does this have any influence on how the task of parsing is defined (see the lecture)? You can find some inspiration in Ranbow (2010): The Simple Truth about Dependency and Phrase Structure Representations. <http://wing.comp.nus.edu.sg/~antho/N/N10/N10-1049.pdf>.
4. Give an example of a sentence with non-projective dependencies.
 - (a) What is the linguistic phenomenon that gives rise to the non-projectivity?
 - (b) Give a pseudo-projective dependency analysis of the same sentence. Show the steps you perform to “projectivize” the tree.
 - (c) Give a dependency analysis of the same sentence in another language (again with universal dependencies), and see if the non-projectivity remains. Add glosses to the non-English words.
 - (d) Do you think the non-projective dependency is essential for downstream applications, or is the projective version good enough? Discuss.
5. We say that a dependency structure is a *dependency forest* if it is (i) acyclic and (ii) single-headed. Show that the following two properties are equivalent within the context of dependency forests:
 - *Projectivity* – defined during the lecture
 - *Planarity* – we say that a dependency forest is *planar* if it can be represented graphically² without crossing edges, provided that:
 - (a) the nodes are laid out horizontally in accordance with the word order,
 - (b) the artificial root node (governing the roots of the connected components of the dependency forest) is placed on the left,
 - (c) the arcs are drawn above the nodes.

¹ V_1 – set of nodes, A_2 – set of labeled arcs

² On a plain