Assignment 4 & 5

Data structures and Algorithms for CL III

Anna Dick, Lea Grüner

February 6, 2021

General tips for your assignments

- write your code as clean as possible
- add comments where useful
- remove unused, redundant or commented out code
- do exactly what the instructions and template specify
- TEST your code under different conditions
- test different input sizes
- test with/without optional arguments specified
- test behavior with unexpected input (ideally no error should be thrown)
- if you don't understand what you're supposed to do, please don't hesistate to ask us (Github issues, tutorial, E-mail)

a4 tips

- check if there are nodes or labels given in the init (assert n or labels not None)
- the basic support functions in 4.1 should be short (only one line if you're using list comprehension)
- Weights can be negative, so don't check if the weight is larger than 0, but if it is not 0.0
- Check all requirements for a tree in is_tree()
- Creating a deepcopy for mst() is not necessary, just create new Digraph with the same nodes and labels
- Try to minimize use of (complicated or unnecessary) helper methods

a5: Projectivity

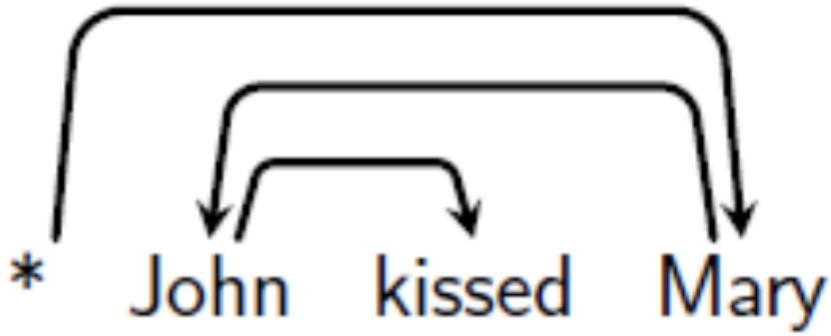
A dependency tree is projective if all edges in the tree are projective, i.e. if there is a directed path from the head word to all the words between the two endpoints of the edge.

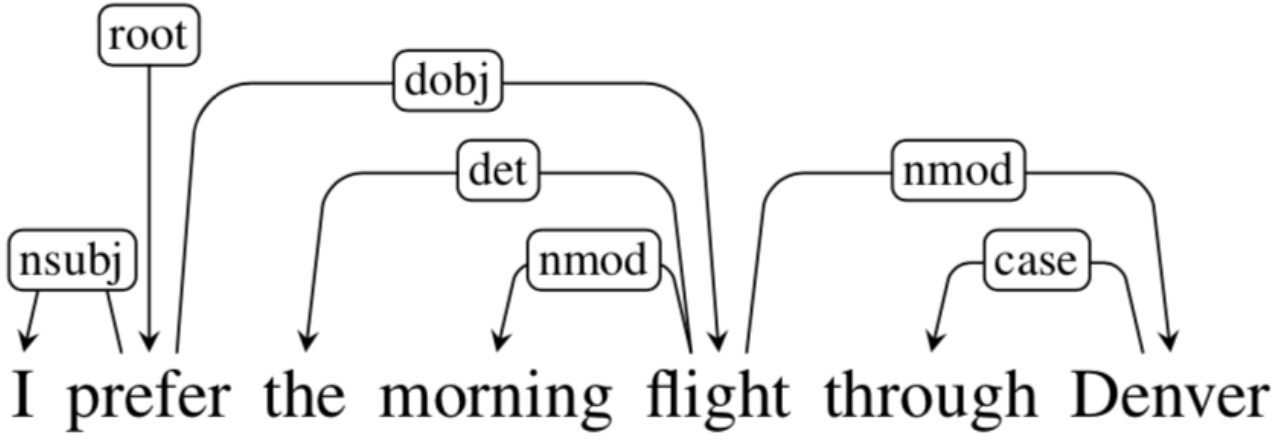
If there is one non-projective edge, the whole tree is non-projective.

Another definition: There are no crossing dependencies in a projective tree.

Much easier to find crossing dependencies (edges where one endpoint lies between the endpoints of another edge and one outside of them)

Projective trees





Non-projective trees