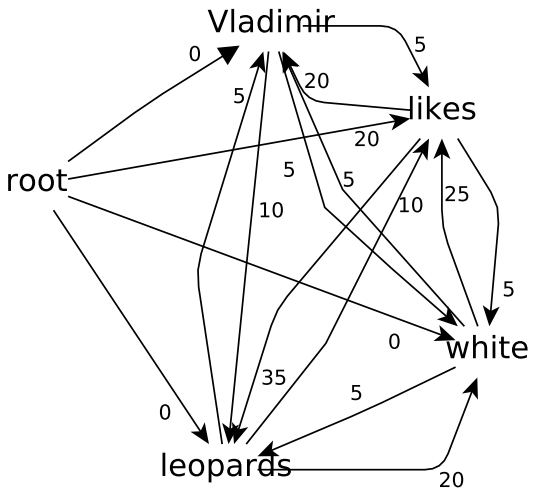
NLP1 2015-16: Assignment 3

1 Dependency parsing / MST

Imagine that you are using an arc-factored non-projective dependency parsing model. You are given a sentence: *Vladimir likes white leopards.*. In this exercise, we consider that there are no labels on arcs (otherwise, recall, you can choose the best one before using the Chu-Liu-Edmonds algorithm). So, the goal is to choose a dependency structure which maximises the sum of arc scores:

$$G = \arg \max_{G \in T(G_x)} \sum_{(i,j) \in G} w_{ij} \tag{1}$$

The scores for each directed arc are given in the following digraph (the table is its weighted adjacency matrix):



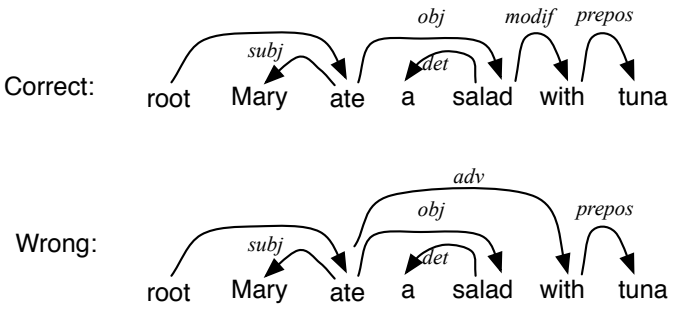
	Heads				
	root	Vlad	likes	white	leopards
Vlad	0	-	20	5	5
likes	20	5	-	25	10
white	0	5	5	-	20
leopards	0	10	35	5	-

1.1 Chu-Liu-Edmonds (25 points)

Explain, step by step, how the CLE algorithm is applied to the given example (similarly to how it was done in our slides) and show what the resulting MST is.

1.2 Arc factorizability (20 points)

In this section we want you to think about the arc factorability assumption (i.e. the assumption that a tree score is sum of arc scores) entails, and how this assumption is likely to affect parsing performance. Consider a sentence *Mary ate a salad with tuna.* Two dependency structures can be considered:



The first one correctly attaches the prepositional phrase *with tuna* to the noun *salad*, whereas the second one attaches it to the verb *ate*. Note that the second structure would be the right one if the sentence is *Mary ate a salad with friends*.

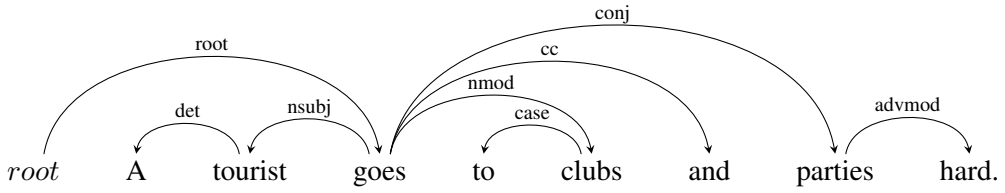
Consider an arc-factored model. Is a basic arc-factored model likely to predict correct trees for both sentences (. . . *with tuna* and . . . *with friends*)? By the basic model here we mean a model which computes the score of an arc w_{ij}^k by looking only in the properties of words at the ends of an arc and at the arc label (i.e. it does not take into account context of the words in the sentence). Explain.

Can you define sentence features for an arc-factored model which would be able to handle this situation (at least to a large degree) without breaking the arc-factorizability assumption? What would they be? Or, do you think that relaxing the assumption (i.e. considering larger fragments than arcs) is the only way to tackle this challenge? Explain / discuss.

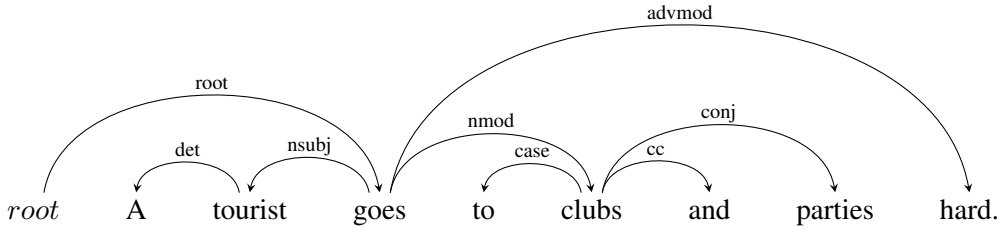
2 Transition-based dependency parsing 35 points

2.1 Derivations

Consider a sentence "A tourist goes to clubs and parties hard.". Assume that a correct dependency tree for this sentence is the following:



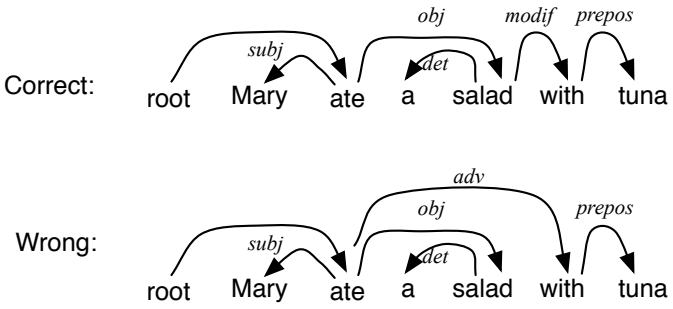
However, assume that a transition-based dependency parser mistakenly predicts the following structure:



List transitions for a transition-based dependency parser (i.e. configurations and decisions made) until the mistake was made by the parser.[1] Explain what was the mistake (something like: in configuration *X* the parser made the decision *W* instead of *Z*). If it is tricky for you to draw that many dependency graphs, it is sufficient to show for each configuration the stack, the buffer, what kind of action was made, and which arcs are added to the arc set A (if any).

2.2 Expressive power

Recall the examples from the previous section:



Last time, you perhaps realised that a 'basic' arc-factored model (see the definition in the assignment) could not disambiguate the two possibilities reliably. What about a transition-based parser? What if you use at least some form of beam search rather than choose a single decision greedily after each decision? Would it be effective in dealing with this type of ambiguity?[2]

Discuss.

[1] There are several ways how dependency trees can be mapped to derivations. Please follow the derivation order defined in the lectures. In literature, this derivation order is known as the *arc-eager version* of the *Nivre's algorithm*.

[2] We mentioned the beam search algorithm during the class, but I have not defined it properly, and perhaps some of you are not familiar with what it is. For the purpose of this question, it is sufficient to understand, that beam search is somewhere in between two extremes: greedy search (where parser selects locally best decision at each step, according to a classifier) and exhaustive search where exponentially many potential derivations are scored. Still, if you are curious: beam-search is an approximate search algorithm which, when applied to history-based parsing, at each parsing step uses k partial derivations provided by the previous step (i.e. k alternative configurations). It considers all potential next decisions (e.g., $LeftArc_r$, $RightArc_r$ for all r, $Reduce$ and $Shift$) for these k partial derivations, and selects k best ones out of this (much larger) set of possibilities. These k best choices constitute k new configurations (new partial derivations) provided to the parser at the next step. The scoring of these possibilities, i.e. (C, d) pairs, where C is a partial derivation and d is a potential next decision, is normally done according to the model, e.g. $P(C, d) = P(C) \times P(d|C)$. If $k = 1$ then we have greedy search (locally best decision at each step) and if $k = \infty$ it is equivalent to (generally intractable) exhaustive search over the exponential space.

3 Comparison of global vs. local models 20 points

Here in this exercise we refer to arc-factored models as global and transition-based parsers as local models. We would like to argue for various factors that affect the accuracy of parsers and compare global and local models in that regard. Among such factors are:

- **Length factors**
    1. sentence length
    2. dependency length
- **Graph factors**
    4. Distance from the sentence root
    5. Degree of non-projectivity
    6. Number of siblings of a node
- **Linguistic factors**
    7. Part of speech
    8. Dependency type (root, subject, object)

For each factor, argue which model is expected to perform better, global or local? Try to put your reasoning based on how each model does inference, training and feature representation.

4 Submission rules

The deadline for the assignment is Sunday, December 13. As before, the assignment may require some pictures, so we are OK with you submitting a hand-written version. You can hand in your assignment during the lab session to a teaching assistant. Alternatively, you can submit a scanned or typed version over email to nlp.uva.2015+hw3@gmail.com with subject *Assignment 3*. In either case the deadline is at **11:00** on December 13.