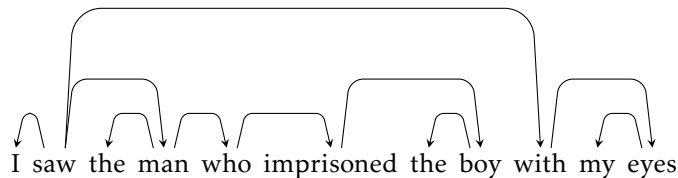


Dependency Parsing exercises: Beam-search, easy-first, dynamic oracles

Deadline: 04.07.2018. Please send completed solutions to jakub.waszczyk@phil.uni-duesseldorf.de with subject "dependency homework".

1. Consider the following unlabeled dependency tree:¹



Let's assume the easy-first parsing strategy. Recall that for an edge to be built, the child must first acquire all its own children.

- Based on your own intuition of which dependencies are easy to determine and which are hard, propose the order in which the arcs should be created by the easy-first parser so as to obtain the above dependency tree.

Is there only one sensible order?

If many, how the easy-first parser is supposed to choose the correct one?

- In the processing order you proposed, which is the "hardest" arc-adding action performed by the parser? Is it indeed performed at the very end of parsing? Discuss.

2. Recall that conventional transition-based (arc-standard, arc-eager) parsers are *greedy* and that *beam search* allows to approach the accuracy of parsers with *exact inference*.

Consider an arc-eager parser with the following, generic scoring function:

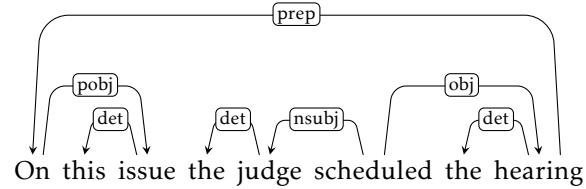
$$score(parse) = \sum_{action \in parse} score(action), \quad (1)$$

where *actions* should be understood as transitions applied to the subsequent parsing configurations.

- What property the parser would have to satisfy in order to be exact w.r.t. this scoring function?
- What beam size would be required to make the beam search extension of the arc-eager parser exact?

¹This analysis, in contrast with Universal Dependencies, assumes that (i) prepositional phrases are headed by prepositions, and (ii) relative clauses are headed by relative pronouns.

3. Consider the following dependence tree:



Let's assume the arg-eager parsing strategy, where:

- The initial configuration is $([], [\text{On, this, ... , the, hearing}], \emptyset)$
- A terminal configuration is any configuration of the form $(\sigma, [], A)$

Answer the following questions:

- What will be the set of arcs if we use the conventional *static oracle* to reconstruct this tree? Does it differ from the set of arcs in the tree shown above? If so, why is this the case?
- What will be the set of arcs if we use the *dynamic oracle* to reconstruct this tree?

Additionally, can you think of dependency trees for which:

- (a) The static oracle would reconstruct a higher number of arcs than the dynamic oracle?
- (b) The static oracle would reconstruct less than a half of the arcs, while the dynamic oracle would retrieve almost all the arcs?