

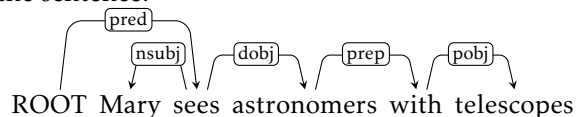
Dependency Parsing exercises:

Transition-based arc-eager parsing, non-projective parsing

Deadline: 26.06.2018. Please send completed solutions to jakub.waszczyk@phil.uni-duesseldorf.de with subject "dependency homework".

1. Arc-eager parsing.

- (a) Enumerate the configurations an arc-eager transition-based parser goes through when parsing the sentence:



A transition is a left-arc, right-arc, shift, or reduce operation (LA, RA, S, R). At each step, indicate the operation, the contents of the stack, the input buffer, and which dependency is added, if any:

- initial state: [ROOT] [Mary sees astronomers with telescopes] \emptyset
 - SH: [ROOT Mary] [sees astronomers with telescopes]
 - LA_{SBJ}: [ROOT] [sees astronomers with telescopes] + (Mary ^{SBJ} ← sees)
 - ...
- (b) Consider the same training treebank, the same machine learning method (nearest neighbor), and the same features (two top words on the stack, two top words in the buffer) as in Ex. 2 last week. However, assume that we use the arc-eager parser instead of the arc-standard parser.
- Will the parser trained on this treebank change the attachment of the preposition from *astronomers* → *with* to *sees* → *with*, as was the case with the arc-standard parser?
 - What feature types would help the parser to learn from this treebank that prepositions always attach to verbs?¹
- (c) Consider the following sentence: *A boy, with a kite playing, went away.* Assume that the parser ignores punctuation. Which of the two strategies, standard or eager, would be better adapted to parse this sentence? What does it tell you about the advantages (or disadvantages) of the arc-eager parser in comparison with the arc-standard parser?
- (d) Which strategy – arc-standard or arc-eager – gives better empirical results according to the following paper?
- <http://aclweb.org/anthology/D14-1082.pdf>

2. Non-projectivity.

- (a) Is the online re-ordering parser, as defined during the lecture, a version of the arc-standard parser or rather the arc-eager parser? Could the other parser also be adapted to perform online re-ordering? If so, how?

¹Of course this is not true in general, only within the context of our small treebank.

- (b) Consider the online re-ordering parser again. Try to determine an example of a “projective order” (as defined during the lecture) such that the parser would have to perform $\mathcal{O}(n^2)$ swaps in order to obtain it (starting from the order $[0, 1, 2, \dots, n]$).
- (c) Following (c), try to find an example of a non-projective dependency tree whose projective order would require $\mathcal{O}(n^2)$ swaps.
The dependency tree can be artificial, i.e., does not have to correspond to any sentence/structure actually occurring in natural languages.
- (d) Does the online re-ordering parser have any theoretical disadvantages in comparison with the k -planar parser? Name at least one. What about the disadvantages of the k -planar parser w.r.t. the online re-ordering parser?
- (e) Based on the results reported in the following papers:
- <http://aclweb.org/anthology/P09-1040.pdf>
 - <http://aclweb.org/anthology/P10-1151.pdf>
- determine if one of these two approaches – online re-ordering and k -planar parsing – is superior when taking into account empirical results.