Dependency parsing

Transition-based parser

Inspired by shift-reduce parser.

Parsing consists of a sequence of transitions between "configurations".

"arc standard" parser

1. Define a "stack" containing <root>, an input "buffer", and an (empty) list of dependency relations.
2. Choose a transition:
    - SHIFT - Remove the word from the front of the input buffer and push it onto the stack.
    - LEFT_ARC - Assert a head-dependent relation between the word at the top of the stack and the word directly beneath it; remove the lower word from the stack.
    - RIGHT_ARC - Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack.
3. Repeat (2) until the buffer and stack are empty.

"arc standard" parser, cont.

Choosing a transition given the current configuration (stack, buffer, relations) is a machine learning problem.

https://nlp.stanford.edu/pubs/emnlp2014-depparser.pdf

https://arxiv.org/abs/1603.04351

https://www.ijcai.org/Proceedings/2019/704

Graph-based parsing

Treat the dependency structure as a directed acyclic graph (DAG) and use tools of graph theory.

Out of all possible DAGs $t \in G_S$ for a rooted sentence S, find the best one: the one with the highest score $\mathrm{score}(t, S)$.

"Edge-factored" models

$$\mathrm{score}(t, S) = \sum_{e \in t} \mathrm{score}(e)$$

MST parsing

a. Construct the fully-connected, directed graph with all possible dependency relations.

b. Weight each edge according to its score.

c. Find the maximum spanning tree (MST)