GWV – Grundlagen der Wissensverarbeitung

Tutorial 6 : Search and Parsing

Exercise 6.1 : (Search and Parsing)

A syntactic structure of a sentence can be described as a dependency tree. Figure 1 shows an example of such a tree. As you can see, every word is attached to another word except "isst", which is the root of the tree, as it's the main verb of the sentence. A parser takes a sentence as input and produces a (dependency) tree as output.

Read the following paper (especially Section 3):

Nivre, J. (2003). An Efficient Algorithm for Projective Dependency Parsing. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03), Nancy, France, 23-25 April 2003, pp. 149-160. `http://stp.lingfil.uu.se/~nivre/docs/iwpt03.pdf`

1. (a) By what operations is the input transformed into the output? What do these operations do?
   (b) When does the parsing algorithm terminate?
   (c) Describe the formal properties of a dependency tree as defined in the paper.
   (d) For each property: Give an example dependency tree that violates the property. Note: We ask for a *tree*, not a sentence! Do not try to find a matching sentence to your trees, it will only distract you.

   (4 Pt.)

2. Try to use the proposed parser actions to produce the tree depicted in Figure 1. Write down the steps and the intermediate states. (2 Pt.)

3. The basic idea laid out in the paper by Nivre is still in use in state-of-the-art parsers such as the one coined "Parsey McParseface" by Google. While the actions are still the same, the mechanics of selecting actions are quite different. The actions are not

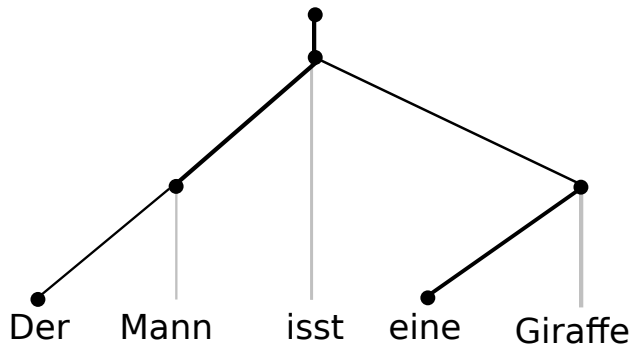

Figure 1: example for a dependency tree

ordered using a fixed precedence and their applicability is not restricted by lexical rules as they are introduced in the paper. Instead, search is used.

If you now view parsing using the proposed actions as a search problem:

- What are the search states?
- What is the start state?
- What are the end states?
- What are the state transitions?
- Can the search space be created before parsing starts?
- What is the advantage of the proposed algorithm in contrast to simply trying to find a good dependency tree by enumerating all possible trees and selecting the best one from them?
- For the search strategies discussed so far: are they a good fit for this search problem and why (not)?
- How would you design a parser using the parser actions together with an appropriate search procedure?

(6 Pt.)

Version: November 16, 2017
Achievable score on this sheet: 12