Dependency Parsing exercises: Implementation of a transition-based parser (part 1)

Deadline: 07.06.2021

Please send completed solutions to `waszczuk@hhu.de` and `evang@hhu.de` with subject "dependency homework" and attachment `parser.py`.

1. Download Matthew Honnibal's simple parser and the Penn Treebank WSJ training data (see course website for links, see Rocket.Chat for password).
2. The parser as is requires a non-standard input format with four columns where the head of each word is given as a 0-based index into the sentence, and the root is indicated by −1. However, the CoNLL-X format has ten columns and uses 1-based indices where the root is indicated by 0. Adapt the parser to use this format. Hint: change the function `read_conll`.
3. Run the parser on the provided data. It requires the following arguments: 1) path to a directory to store the model in, 2) path to the training data (WSJ section 02-21) in CoNLL-X format, 3) path to the validation data (WSJ section 22) in part-of-speech-tagged format, 4) path to the validation data in CoNLL-X format. With 15 training iterations, the process takes 46 minutes on my laptop, the final accuracy on the validation data should be 90.09.
4. The command-line interface of the parser could do with some cleaning up. First, note that the given model directory is created, but the model files are actually just stored in the current directory. Fix that so that they are actually stored in the model directory.
5. Secondly, note that the third argument is superfluous. The gold parts of speech of the validation data are not used – that would be a bad practice, as real-life input data does not come part-of-speech-tagged. Also, the input tokens can just as easily be read from the CoNLL-X input. Remove the third argument and change the code accordingly.
6. Thirdly, the parser can currently only be invoked in training mode, where it trains a model on the training data, saves it, and reports its accuracy on the validation data. Add a way to invoke it in parsing mode, where you give it a previously trained model and some tokenized sentences without any annotation, and it outputs parses for these sentences.
7. Train the parser again and run it on some input to see if everything (still) works.

Hints

- The parser is written in Python 2, so you need to use that interpreter/configure your IDE accordingly – or convert everything to Python 3.