1 Introduction

In the previous exercise, we extracted D-rules from a Swedish treebank. In this exercise, we are going to use these D-rules to guide a transition-based parser using the arc-standard algorithm for dependency parsing.

2 Arc-standard dependency parsing

Starting from a predefined initial configuration, a transition-based dependency parser repeatedly applies transitions (or parser actions) proposed by a guide until it reaches a terminal configuration. In the arc-standard parsing algorithm, there are three types of possible transitions:

1. Shift: Move the next word from the buffer to the stack.
2. LeftArc-L: Add a left-directed arc with label \( L \) from the word on top of the stack to the secondmost word on the stack and remove the latter from the stack.
3. RightArc-L: Add a right-directed arc with label \( L \) to the word on top of the stack from the secondmost word on the stack and remove the former from the stack.

The guide is normally constructed using machine learning, but in this exercise we are going to use D-rules together with a simple heuristic.

3 Run the parser

The program `parser.py` contains an implementation of arc-standard dependency parsing using a set of D-rules like the ones we constructed in the previous exercise. The core of the parser is in the following code, where the variable `sentence` holds (the indices of) the words of the sentence to be parsed:

```python
stack = []
buffer = sentence
while buffer != [] or len(stack) > 1:
    if len(list(stack)) < 2:  # Shift if less than two words on stack
        stack.append(buffer[0])
        buffer = buffer[1:]
    elif (tag[stack[-1]], tag[stack[-2]], "<") in drules:  # LA-L if D-rule
        head[stack[-2]] = stack[-1]
        dep[stack[-2]] = drules[(tag[stack[-1]], tag[stack[-2]], "<")][0]
        stack = stack[:-2] + stack[-1:]
    elif (tag[stack[-2]], tag[stack[-1]], ">") in drules:  # RA-L if D-rule
        head[stack[-1]] = stack[-2]
        dep[stack[-1]] = drules[(tag[stack[-2]], tag[stack[-1]], ">")][0]
        stack = stack[:-1]
    elif buffer != []:  # Shift if words left in buffer
        stack.append(buffer[0])
        buffer = buffer[1:]
    else:  # Pop the stack otherwise
```

The parser is initialized to an empty stack and a buffer containing all the words of the sentence. It then keeps applying transitions as long as the buffer is not empty or the stack has more than one element. The guide is implemented as an if-then-else clause that always shifts if there are less than two words on the stack, else adds a left-arc if this is licensed by a D-rule, else adds a right-arc if this is licensed by a D-rule, else shifts if there are words left in the buffer, and otherwise just pops a word off the stack to make progress towards a terminal configuration. In order to run the parser, you should run a command like the following:

```bash
python parser.py 0 drules.txt sv-universal-test.conll > out.conll
```

The first argument is a frequency threshold for the D-rules; only rules with a frequency greater than the threshold will be considered by the parser. The second argument is the file containing the D-rules, and the third argument is the test file in CoNLL-X format. The parser writes the output to stdout in CoNLL-X format.
4 Evaluate the parser
You can evaluate the parser using `eval.pl` as follows:

`./eval.pl -g sv-universal-test.conll -s out.conll -q`

Try using different frequency thresholds for the D-rules and see how this affects the results. Can you explain what is happening?

5 Analyze the results
Go through 5–10 sentences in the test set and try to analyze the errors performed by the parser. Pick a reasonably short sentence (with some parsing errors in it) and see if you can figure out the correct sequence of transitions needed to derive the correct tree. Check in your rule file if all the necessary D-rules can be found in the training set. Discuss how the parsing process could be improved to increase the parsing accuracy.

6 Submitting the third assignment
The fourth assignment of the course consists of your work with Exercise 7 and 8. Make sure that you submit all of the following to joakim.nivre@lingfil.uu.se at the end of the week:

- The D-rules extracted from the training set.
- A short report of the evaluation results with different frequency thresholds, and an error analysis including the correct transition sequence for at least one sentence and a discussion of possible improvements.