

# Dependency Parsing: Error Analysis

Deadline: 11.06.2018. Please send completed solutions to [jakub.waszczyk@phil.uni-duesseldorf.de](mailto:jakub.waszczyk@phil.uni-duesseldorf.de) with subject "dependency homework" and an attachment "ex7\_lastname(s).pdf".

## 1 Introduction

The goal of this session is to perform *error analysis* of the arc-factored model we trained on 15 May. That is to say, we will try to determine what types of errors the parser performs frequently, and how to improve the parser in order to deal with them.

Download the archive accompanying the today's session. It is based on the source code and files we used on 15 May. Place the archive file in the home directory, open a `terminal`<sup>1</sup> session, and perform the following steps:

```
$ cd ~
$ unzip error-analysis-package.zip
$ cd error-analysis-package
```

## 2 Training the model (reminder)

**NOTE:** you do not have to repeat training, the `model.txt` file is already available in the package.

To recall, the steps<sup>2</sup> to train the model on sentences of length  $\leq 40$ :

```
$ mkdir train_data
$ ./source/main.py dump -i ud-data/en_partut-ud-dev.conllu -o train_data -l 40
$ ./third-party/estimate-params sgd -t train_data -m model.txt -i 10
```

## 3 Parsing and evaluation (reminder)

To recall, you can use the newly created model to parse sentences in the CONLLU format:

```
$ ./source/main.py parse -m model.txt \
-i ud-data/en_partut-ud-test.conllu -o test.system.conllu
```

To evaluate the quality of parsing:

```
$ ./evaluation_script/conll17_ud_eval.py ud-data/en_partut-ud-test.conllu \
test.system.conllu --verbose
```

As we perform unlabeled parsing, we are mainly interested in UAS – *unlabeled attachment score* – defined as the proportion of words for which the parser correctly assigned the head.

---

<sup>1</sup>Search for the application called *terminal*.

<sup>2</sup>These steps are explained in the tutorial file from 15 May.

## 4 Error analysis

To perform error analysis, you can use the `MaltEval.jar` tool available in the package:

```
$ java -jar -Xmx2g third-party/MaltEval.jar \  
-g ud-data/en_partut-ud-test-no-comment.conllu \  
-s test.system.conllu -v 1
```

The above command allows to visually compare the original (*gold*) test file<sup>3</sup> with the version determined by the parser – `test.system.conllu`.

**Exercise 1.** Determine some types of mistakes frequently made by the parser. Keep in mind that the parser is based on the arc-factored model and that it relies on the feature types defined in the `source/feature.py` source code file. Focus on short sentences and those for which the parser commits a small number of mistakes, e.g.:

(12) “I hope my proposal will be taken into consideration in tomorrow ’s vote .”

(17) “Mr President , Commissioner , I would like to thank Mrs Schroedter for an excellent report .”

(35) “If momma is not happy , nobody is happy .”

(50) “The Council shall act unanimously after consulting the European Parliament .”

**Exercise 2.** Determine which of these mistakes could be avoided by modifying the feature types in the `source/feature.py` source code file, and which would require a more powerful model (with vertical/horizontal Markovization, information about arity, etc.). For the former ones, propose features types which would potentially solve the problems.<sup>4</sup>

---

<sup>3</sup>The *no-comment* version is just a version without comments, which are not supported by `MaltEval.jar`.

<sup>4</sup>You can have a look at <http://www.seas.upenn.edu/~strctlrn/bib/PDF/dependencyACL2005.pdf>, Sec. 2.4, for inspiration on what feature types can be useful in arc-factored parsing.