

MaltEval: An Evaluation and Visualization Tool for Dependency Parsing

Jens Nilsson* and Joakim Nivre*[†]

*Växjö University, School of Mathematics and Systems Engineering, Sweden

[†]Uppsala University, Dept. of Linguistics and Philology, Sweden
{jni, nivre}@msi.vxu.se

Abstract

This abstract presents an freely available evaluation tool for dependency parsing. It is flexible and extensible, and provides functionality for both quantitative evaluation and visualization of dependency structure.

1. Introduction

Dependency parsing in natural language processing has become popular in recent years, both within the NLP research community and as a component in various NLP systems. As a consequence, the CoNLL shared tasks of 2006 (Buchholz and Marsi, 2006) and 2007 (Nivre et al., 2007) was focusing on multilingual dependency parsing using dependency based treebanks, something that has led to a further increase in the popularity of dependency-based parsing methods.

An important component in the two CoNLL shared tasks and in many NLP applications is evaluation, both quantitative and qualitative. There are very few widely used quantitative evaluation software tools for dependency parsing. One exception is `eval.pl`,¹ the evaluation script in Perl created by the organizers of the first CoNLL shared task. It was used for evaluating the output of the participants' parsers, mainly using the quantitative metric labeled attachment score (LAS). Using the same evaluation software is a way of guaranteeing that different systems are compared fairly.

The script `eval.pl` also provides functionality for a more detailed error analysis in a quantitative fashion, such as computing accuracy for individual part-of-speech tags, and precision and recall for individual dependency types, arc depth and arc direction, etc. However, in order to get a deeper understanding of the errors that a parser makes, investigating the errors of individual sentences is also important, as this can be part of the process of improving the parser. This type of qualitative error analysis can be facilitated by visualizing the dependency structure. The script `eval.pl` does not have this functionality, making it unsuitable for this type of qualitative error analysis.

However, visualizing the dependency structure is a functionality that several NLP software tools provide. Some examples are Tree Editor (TrEd) (Hajič et al., 2001), which is a graphical editor and viewer of trees and annotation tool written in Perl. Another visualization tool is NetGraph (Mírovský, 2006), a tool for searching in treebanks in the FS format, the format used for encoding e.g. Prague Dependency Treebank. None of these have support for the data format used in the shared tasks, the CoNLL format,

which has become a de facto standard format for parsing dependency structure.

A visualization tool that supports the CoNLL format is DepSVG (Kaljurand, 2006). However, it only has the ability to produce vector-based image files, one file for each dependency graph. This makes it tedious to use for browsing between the dependency graphs of different sentences. Another drawback of these visualization tools is that they do not provide functionality for quantitative evaluation like `eval.pl`. Moreover, none of the tools can visually compare the parse trees of the gold-standard and the output of a parser by highlighting mismatches, a useful functionality for manual qualitative visual error analysis.

2. MaltEval 1.0

MaltEval is a new software tool written in Java that combines quantitative and qualitative evaluation in one tool. It is to a large extent adapted to the evaluation script `eval.pl`, as the functionality of MaltEval is essentially a superset of this script (when used in the most verbose mode).² It is also more flexible and contains many additional features.

Like the above mentioned visualization tools, MaltEval is able to visualize dependency structure. Unlike them, it provides visual support for qualitative evaluation by highlighting errors.

2.1. Quantitative Evaluation

Here is a list of features that are implemented in MaltEval, but are lacking in `eval.pl`³:

Flexibility MaltEval comes with default evaluation settings, which can be manipulated through flags or files containing the evaluation settings. For instance, MaltEval is executed with default settings like this:

```
java -jar MaltEval.jar -s parser.conll -g gold.conll
```

The same evaluation can also be achieved by

```
java -jar MaltEval.jar --GroupBy Token --Metric LAS  
-s parser.conll -g gold.conll
```

¹And `eval07.pl`, the evaluation script of the shared task 2007 containing minor modifications compared to `eval.pl`

²Currently, the statistics under the heading "Local contexts involved in several frequent errors" in `eval.pl` cannot be produced, and some other information is presented differently.

³Everything is described in more detail in the User Guide that is shipped with the MaltEval distribution.

where the grouping strategy (see below) and the type of metric instead are explicitly specified using flags. Another equivalent way of producing the same evaluation is like this:

```
java -jar MaltEval.jar -e eval.xml
                        -s parser.conll1 -g gold.conll1
```

if the file eval.xml contains:

```
<evaluation>
  <parameter name="Metric"><value>LAS</value>
</parameter>
  <parameter name="GroupBy"><value>Token</value>
</parameter>
</evaluation>
```

Any one of the more than 25 parameters can be specified using either flags or in an evaluation file. The result is by default written to standard output, or to a file by specifying an output file.

Several parsed files MaltEval can not only evaluate single parsed dependency files, but also automatically evaluate multiple files as well as automatic evaluation of cross-validation experiments, including means and standard deviation. The script eval.pl is only limited to evaluating a single file at a time.

File formats Supports a number of dependency-based file formats, such as the CoNLL and MaltXML formats.

GroupBy MaltEval has a large number grouping strategies of the tokens. Besides the default attachment score evaluation (correct tokens / number of tokens), there are currently 16 grouping strategies, such as ArcLength, ArcDepth, BranchingFactor, ArcProjectivity and Frame. Essentially, all the detailed results that eval.pl can produce can also be produced by using the appropriate grouping strategy, specified using the `--GroupBy` flag.

For instance, grouping by dependency label could produce something like this

```
Metric-> LAS
GroupBy-> Deprel

=====
precision    recall      Deprel
-----
0.618        0.488      Row mean
21           24       Row count
-----
-            0      AdvAtr
0.5          0.091   Apos
0.809        0.802   Atr
0.5          0.333   AtrAdv
-            0      AtrAtr
0.556        0.484   Atv
-            0      AtvV
...
```

where the precision and recall are displayed for all dependency labels in the input files. The rows labeled Row mean shows the mean of all dependency labels and Row count the number of distinct dependency labels that appeared in the parsed data (under precision) and in the gold-standard file (under recall). MaltEval is also able to compute other types of attributes such as F-score.

Metric Specify whether to evaluate labeled or unlabeled attachment score or label accuracy, as well as other metric strategies such as error rate for the values of the head and dependency label.

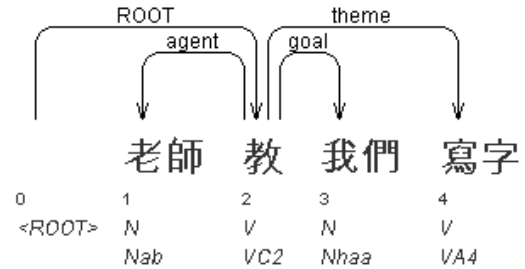


Figure 1: Chinese sentence visualized using MaltEval

Exclude Sentence by Length Support for excluding sentences longer and/or shorter than specified lengths.

Exclude Tokens by Attribute Support for excluding various tokens from the evaluation based on e.g. word form or part-of-speech.

Formatting Flexibility Flexibility to format the evaluation output, such as disabling or enabling detailed output.

Confusion Matrix Possibility to automatically produce confusion matrices for any grouping strategy. Here is an example of such a confusion matrix:

```
Confusion matrix for ArcDirection
left   right   to_root   Col: system / Row: gold
-----
-       1581    64        left
1033    -       60        right
359     245    -         to_root
```

Another confusion matrix will be displayed if one simply changes the grouping strategy to e.g. Deprel, showing how the parser makes mistakes in assigning the dependency types.

Statistical Significance Possibility to automatically test statistical significance if one gold-standard file and two or more parsed file are specified. A piece of the statistically significant result could look like this

```
<1>  <2>  <3>  McNemar: p<0.01?
-----
-      1      0  <1> (parser1.conll1)
-      -      1  <2> (parser2.conll1)
-      -      -  <3> (parser3.conll1)
```

where McNemar's test has been used pairwise between three parsed files for LAS (1=there is a statistically significant difference for $p < 0.01$).

Batching Support for batched evaluation, i.e. possibility to perform several types of evaluations in sequence as in eval.pl, but with a greater flexibility of including and excluding of different type of evaluation settings.

2.2. Qualitative Evaluation

The integrated visualizing module is enabled by simply switching on the visualization flag, which create a window with possibility to browse though the dependency graphs. The visualization of the dependency structure in MaltEval looks deliberately not the same as TrEd, NetGraph and SVGDep. An example is shown in figure 1. It also contains the information that is present in the CoNLL format, either

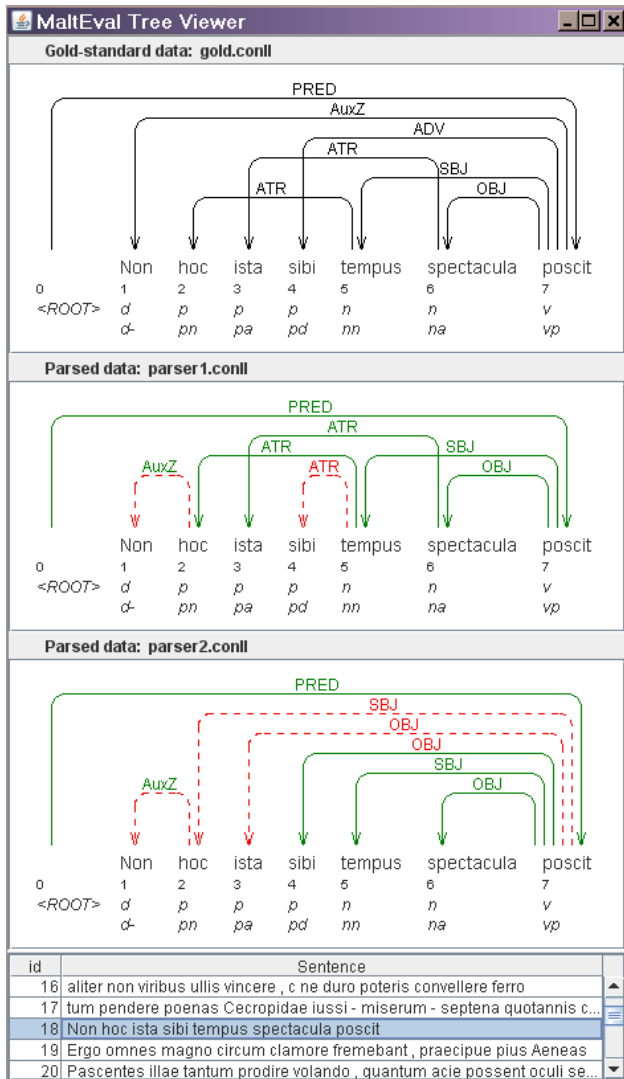


Figure 2: Visualizing many parsed dependency files, and highlighting errors.

in the picture directly or as pop-up labels when the mouse pointer is located over the tokens.

The example also illustrates that MaltEval has full support for Unicode, as the sentence in the particular example is a Chinese sentence displayed using a Chinese font. MaltEval will investigate the tokens of the input file, and as long as there is at least one appropriate font installed on the computer, MaltEval will choose one of these fonts.

In case the dependency graph contains non-projectivity, such as in the top dependency graph of figure 2, it will contain crossing arcs (which might make it easier to spot them than in the dependency structure of TrEd, NetGraph and SVGDep, since non-projectivity produces crossing arcs in MaltEval).

Figure 2 also illustrates how mismatches are highlighted in the parsed file(s) in order to easier perform a manual error analysis. Mismatches are highlighted using both colors (red labels and arcs) and dashed lines for arcs and dependency labels individually.

2.3. Plug-ins, API and Javadoc

Yet another flexible property of MaltEval is the possibly for users with knowledge in Java to write their own grouping strategies. This could be very handy if a user needs one type of grouping strategy that is currently not implemented in MaltEval. The User Guide, that contains information about how to implement, compile and archive a plug-in, comes with the distribution of MaltEval. A comprehensive Javadoc for the MaltEval API is also provided, something that is necessary for developers of plug-ins. A developer must essentially only implement a Java interface that all grouping strategies must do, which is also explained in details in the User Guide.

3. Conclusion

We present an evaluation tool for dependency parsers that combines quantitative evaluation and visualization, which is a property that no tools that we are aware of have. It is flexible in the sense that it comes with a large number of parameters that are easy to modify. It is also flexible and extensible in the sense that new grouping strategies can be plugged into MaltEval even without access to the source code of MaltEval. It is freely available for download and use, but it comes with no guarantees.⁴

4. References

- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*.
- Jan Hajič, Barbora Vidová-Hladká, and Petr Pajas. 2001. The Prague Dependency Treebank: Annotation Structure and Support. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 105–114. University of Pennsylvania, Philadelphia, USA.
- Kaarel Kaljurand. 2006. DepSVG. URL: <http://www.ifi.unizh.ch/cl/kalju/download/depsvg/> (31 March, 2006).
- Jiří Mirovský. 2006. Netgraph: a Tool for Searching in Prague Dependency Treebank 2.0. In *Proceedings of The Fifth International Treebanks and Linguistic Theories conference, Prague, Czech Republic*, pages 211–222.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

⁴The URL to the web-page of MaltEval will be placed here if the abstract is accepted.