

Interuniversity Master's Degree in Artificial Intelligence

Natural Language Understanding – Academic Year 2025-2026

Practical Assignment 2 – Dependency parsing

In this assignment, you will implement a neural parser to output the dependency syntactic structure (in CoNLLU format) of an input given sentence. Figure 1 shows an example of a dependency tree:

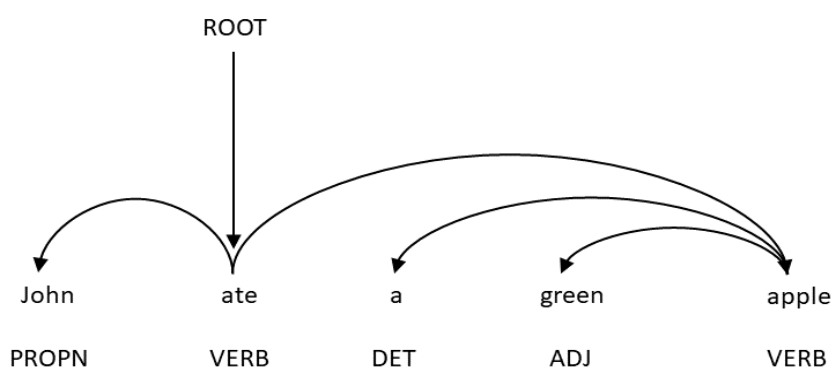


Figure 1. Example of an input sentence ('John ate a green apple') and a valid dependency structure.

More particularly, you are asked to implement a transition-based dependency parser based on feed-forward networks.

For the transition-based algorithm we will use the arc-eager algorithm. The set of transitions and preconditions are described in Table 1. In case you need it, more information about this transition-based algorithm can be found at: <https://aclanthology.org/J08-4003.pdf> (Section 4.2).

Transitions	Current state	Next State
LEFT-ARC _i	$(\sigma i, j \beta, A)$	$(\sigma, j \beta, A \cup \{(j, i, i)\})$
RIGHT-ARC _i	$(\sigma i, j \beta, A)$	$(\sigma i j, \beta, A \cup \{(i, i, j)\})$
REDUCE	$(\sigma i, \beta, A)$	(σ, β, A)
SHIFT	$(\sigma, i \beta, A)$	$(\sigma i, \beta, A)$

Preconditions	
LEFT-ARC _i	$\neg[i = 0]$ $\neg \exists k \exists l [(k, l, i) \in A]$
RIGHT-ARC _i	$\neg \exists k \exists l [(k, l, j) \in A]$
REDUCE	$\exists k \exists l [(k, l, i) \in A]$

Initial state	Final state
$(\sigma = [\text{ROOT}], \beta = \text{sentence}, \{\})$	$(\sigma, \beta = [], A)$

For the neural model, we propose a base architecture. The model must receive as input a sequence of features (words and /or universal PoS tags) from the n top elements in the stack and the buffer. Each feature will be mapped to an embedding, and the concatenations of those embeddings will serve as input to a Dense Layer, followed by an output layer with a softmax activation layer. You have the freedom to modify the dimensions of the network, explore hyperparameters to improve the performance, and expand the base model. Figure 2 shows the high-level architecture of the model.

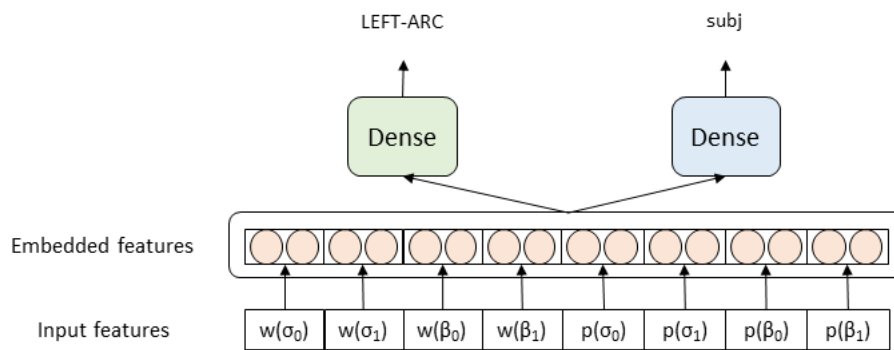


Figure 2. High-level architecture of the model, using as input features the top 2 words in the stack and the buffer. W is a function that retrieves the word form from a token and P is a function that retrieves the part-of-speech that from a token.

The functionalities that must be supported are:

1. **Train a model**, and verify that converges successfully on the development set.
2. **Once your model is trained**, implement a function to predict the outputs for the test set and report the standard metrics for parsing: labelled (LAS) and unlabeled attachment scores (UAS). For that, you must use the official evaluation script used in universal dependency parsing: https://universaldependencies.org/conll18/conll18_ud_eval.py

Note: As you know from the P1 assignments, the conllu training, development and test splits contain lines corresponding to multi-word expressions and empty tokens (encoded with IDs of the form X.Y or X-Y), which you were asked to ignore. In this assignment, we will also proceed this way. In particular, you will work with a ‘cleaned’ version of these conllu files that do not contain such lines, which will simplify the use of the conll18_ud_eval.py file.

Specifically, for this assignment you must train and evaluate your model on the [UD English-ParTUT](https://universaldependencies.org/) dataset from the <https://universaldependencies.org/> (UD) collection, using the corresponding training, development and test files, represented in CoNLL-U format (the details about this format can be found at: <https://universaldependencies.org/format.html>). The UD English-ParTUT dataset contains some non-projective sentences (sentences with crossing arcs) that cannot be modeled by the arc-eager algorithm and therefore must be removed (the code for this is already provided in materials/conllu_reader.py). Additionally, once you apply the dependency parser you developed, there may be some arcs predicted in a way that produces an ill-formed tree. Before proceeding with the evaluation using the conll18_ud_eval.py file, you will need to repair the tree (the code for this is already available in materials/postprocessor.py).

For more details about the assignment, you can also check the file `p2_guide.pdf`.

Submission

- The assignment must include a short user manual that indicates how to run the code for training, evaluation, and generation of labels. It must also include a brief discussion of the implementation decisions, differences across the evaluated models that you might have explored, as well as an analysis of the performance across the evaluated datasets. The document must not exceed 3 pages, and it will be written using Calibri font style and a size of minimum 11pt.
- The assignment will be done in groups of 2 or 3 people.
- The assignment will be submitted to the virtual campus, on the 9th of December at 15:30 the latest. Only one member of the group will submit the assignment. Assignments submitted after the deadline will be automatically considered failed.
- The assignment will be defended during the laboratory hours or during tutoring sessions specifically arranged for assessment if required so. The slots will be made available by the professor at the time. Both members of the group must be present during the defense, otherwise the missing member(s) will fail the assignment with a score of zero.
- This assignment accounts for 25% of the overall course grade.