Interuniversity Master's Degree in Artificial Intelligence

Natural Language Understanding – Academic Year 2022-2023

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:

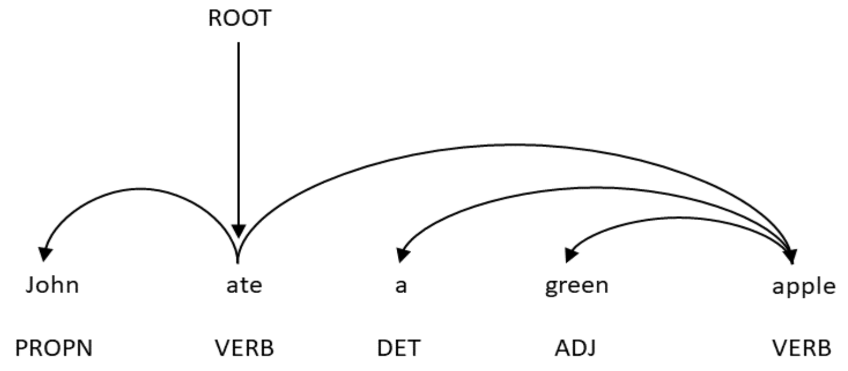

Figure 1. Example of an input sentence ('John ate a green apple') and a valid dependency structure.

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$_l$	(σ|i, j|β, A)	(σ, j|β, A ∪ {(j, l, i)})
RIGHT-ARC$_l$	(σ|i, j|β, A)	(σ|i|j, β, A∪ {(i, l, j)})
REDUCE	(σ|i, β, A)	(σ, β, A)
SHIFT	(σ, i|β, A)	(σ|i, β, A)

Preconditions	
LEFT-ARC$_l$	¬[i = 0] ¬∃k∃l [(k, l , i) ∈ A]
RIGHT-ARC$_l$	¬∃k∃l [(k, l , j) ∈ A]
REDUCE	∃k∃l[(k, l, i) ∈ A]

Initial state	**Final state**
(σ =[ROOT], β=sentence, {})	(σ, β = [], 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.

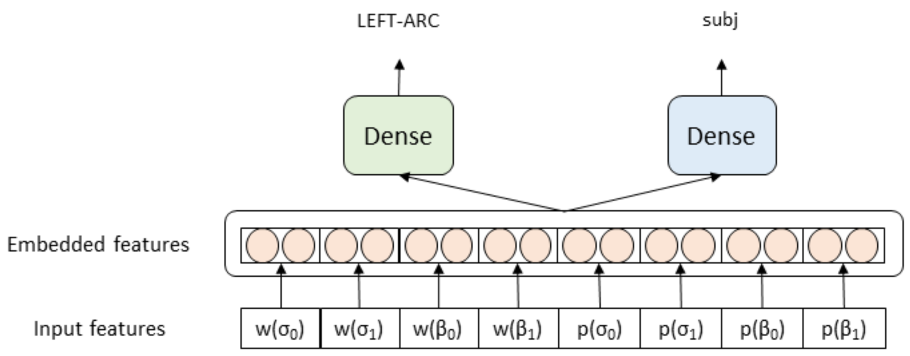

Figure 2. High-level architecture of the model, using as input features the top 2 words in the stack and the buffer. W is a function that retrieves the word form from a token and P is a function that retrieves the part-of-speech that from a token.

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 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 that have crossing arcs), which cannot be modelled by the arc-eager algorithm. We provide in Appendix 1 a snippet of the code to identify if a sentence is projective, in order to remove non-projective *sentences from the training and development sets only*.

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 people.

- The assignment will be submitted to the virtual campus, on the 18th of December at 22.00 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 the week after the submission, during the laboratory hours. 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.

Appendix 1

```
def is_projective(arcs: list):
"""
Determines if a dependency tree has crossing arcs or not.

Parameters:
    arcs (list): A list of tuples of the form (headid, dependentid, coding
the arcs of the sentence, e.g, [(0,3), (1,4), …]

Returns:
    A boolean: True if the tree is projective, False otherwise
"""


  for (i,j) in arcs:

    for (k,l) in arcs:

      if (i,j) != (k,l) and min(i,j) < min(k,l) < max(i,j) < max(k,l):
        return False

  return True
```