1a)

|  |  |  |  |
| --- | --- | --- | --- |
| stack | buffer | New dependency | Transition |
| *[root]* | *[Nadia, rode, the, old, donkey, with, dexterity]* |  | *Initial config* |
| *[root, Nadia]* | *[rode, the, old, donkey, with, dexterity]* |  | *shift* |
| *[root, Nadia, rode]* | *[the, old, donkey, with, dexterity]* |  | *shift* |
| *[root, rode]* | *[the, old, donkey, with, dexterity]* | *rode → Nadia*  *(nSubj)* | *left-arc* |
| [root, rode, the] | [old, donkey, with, dexterity] |  | shift |
| [root, rode, the, old] | [donkey, with, dexterity] |  | shift |
| [root, rode, the, old, donkey] | [with, dexterity] |  | shift |
| [root, rode, the, donkey] | [with, dexterity] | donkey → old  (amod) | left-arc |
| [root, rode, donkey] | [with, dexterity] | donkey → the  (det) | left-arc |
| [root, rode] | [with, dexterity] | rode → donkey  (dobj) | right-arc |
| [root, rode, with] | [dexterity] |  | shift |
| [root, rode, with, dexterity] | [] |  | shift |
| [root, rode, with] | [] | with → dexterity  (pobj) | right-arc |
| [root, rode] | [] | rode → with  (prep) | right-arc |
| [root] | [] | root → rode | right-arc |

1b)

A sentence containing n words will be parsed in exactly 2n steps. For each of the n items, it will take n steps to shift it into the stack and n steps to remove it from the stack with either a left-arc or right-arc.

1c)

We will go through the shift-reduce parse, as we did for part 1a:

|  |  |  |  |
| --- | --- | --- | --- |
| stack | buffer | New dependency | Transition |
| [root] | [John, saw, a dog, yesterday, which, was, a Yorkshire Terrier] |  | Initial config |
| [root, John] | [saw, a dog, yesterday, which, was, a Yorkshire Terrier] |  | Shift |
| [root, John, saw] | [a dog, yesterday, which, was, a Yorkshire Terrier] |  | Shift |
| [root, saw] | [a dog, yesterday, which, was, a Yorkshire Terrier] | saw → John  (nsubj) | left-arc |
| [root, saw, a dog] | [yesterday, which, was, a Yorkshire Terrier] |  | Shift |
| [root, saw, a dog, yesterday] | [which, was, a Yorkshire Terrier] |  | Shift |

At this point in the reducer, it would’ve been ideal to remove ‘yesterday’ from the stack with the dependency (*saw* → *yesterday)*, however a dependency can only be made with the two most recent items added to the stack. Due to *‘saw*’ being the 3rd most recent item in the stack, we are unable to reach this dependency. I will continue the shift-reducer algorithm to see the result of it - to further illustrate what this causes for a non-projective dependency tree:

|  |  |  |  |
| --- | --- | --- | --- |
| stack | buffer | New dependency | Transition |
| [root, saw, a dog, yesterday, which] | [was, a Yorkshire Terrier] |  | shift |
| [root, saw, a dog, yesterday, which, was] | [a Yorkshire Terrier] |  | shift |
| [root, saw, a dog, yesterday, was] | [a Yorkshire Terrier] | was → which  (nsubj) | left-arc |
| [root, saw, a dog, yesterday, was, a Yorkshire Terrier] | [] |  | shift |
| [root, saw, a dog, yesterday, was] | [] | was → a Yorkshire Terrier  (attr) | right-arc |

At this point, we are unable to assign the dependency (a dog → was) due to ‘a dog’ being the third item in the stack. With the crossing edge here, this non-projective dependency tree is unable to use the parsing mechanism.

Report for 2b

By the end of the 10th epoch, I was able to successfully get a loss of 0.109 - less than the recommended 0.11. I was also able to get a LAS score of 0.886 and a UAS score of 0.901. I was consistently hovering around ~0.85 for the LAS score until I passed the self.training as an argument for the dropout function:

h\_drop = F.dropout(h, self.config.dropout)

Vs.

h\_drop = F.dropout(h, self.config.dropout, training=self.training)

Overall, I found implementing get\_oracle the most challenging part of the assignment due to the corner cases that needed to be considered.