ASSIGNEMENT 1 REPORT
NLU

Francesco Ferrini
ID: 224010

Exercise 1:
<u>Define a function to extract a path of dependency relations from the ROOT to a token</u>

In the **getDependency** function a sentence is given as input. Then it is parsed getting a Doc object of spacy (**doc**). The for loop iterates over all the tokens in the sentence. For each token is instantiated a list (**list1**) where the dependencies will be inserted. In the second for we iterate over the ancestors of the token taken into consideration. For every ancestor in the list will be inserted his dependency and the text related to him (**anc.text**) in the list. Since we want that the root is in the first position we have to reverse the list. Then we insert the token dependency and the token text at the end of the list in order to have the complete path. In that way in every list in output we obtain the path from the root to a given token.

Exercise 2:
<u>Extract subtree of a dependents given a token</u>

In the **subtreeExtraction** function a sentence is given as input. Then it is parsed getting a Doc object of spacy (**doc**). The for loop iterates over the tokens in the parsed sentence. For every token is instantiated a list (**list1**) where we will insert the subtree related to the token taken into consideration. Then we extract the subtree (**token.subtree**).
In the second for loop we insert in the list every text of the subtree tokens checking to not add also the token text (since is asked to insert only the dependents).

Exercise 3:
<u>Check if a given list of tokens (segment of a sentence) forms a subtree</u>

In the **checkSubtree** function a sentence and a list of words is given as input. The for loop iterates over all the tokens of the parsed sentence and extracts the subtrees. Then the if checks weather the list of words given as input is one of the subtrees. If yes the function returns True, otherwise False.

Exercise 4:
<u>Identify head of a span given its token</u>

In the **spanHead** function a span is given as input where a span is a slice of a sequence. Simply using span.root we can easily find the the token with the shortest path to the root of the sentence and so the head of the span.

Exercise 5:
<u>Extract sentence subject, direct object and indirect object spans</u>

In the **extractSpanInfo** function a sentence is given as input. Firstly a dictionary with 3 keys is created. The 3 keys are strings: “subj”, “dobj”, “iobj”. For every key, is created a list. In the list corresponding to the key “subj” are inserted all the words that forms a subject span (I used both nominal subject and nominal subject passive since they are both subject). In the list corresponding to the key “obj” are inserted all the words that form a direct object span and in the list corresponding to the key “iobj” are inserted all the words that form an indirect object span.