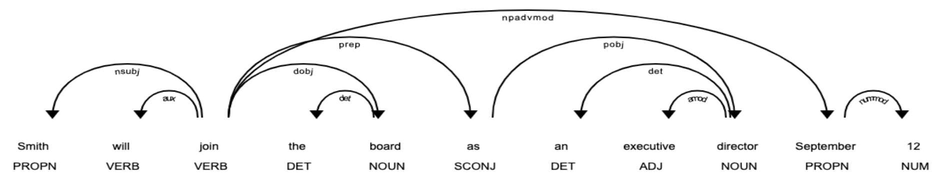
# Report

## Assignment: Working with Dependency Graphs (Parses)

The objective of the assignment is to learn how to work with dependency graphs by defining functions.

SENTENCE: Smith will join the board as an executive director September 12

**Let's See Dependencies Graph:**



## 1.Extract a path of dependency relations from the ROOT to a token

Dependency-based approaches to syntactic parsing assume that the syntactic structure of a sentence can be analysed in terms of binary dependency relations between lexical units, units that in the simplest case are taken to correspond directly to the tokens of the sentence.

Here I am extracting the path of dependency relations from the Root to a token. In sentroot the sentence is first read and then passed to extractRootPath function over looping to store the list of tokens with root of the word {t.text: extractRootPath(t, doc) for t in doc}
In extractRootPath for each token in doc jumping from a token to its head until the root is reached. If the dependency is not equal to the root, then adding the token at the start of the list. The output is saved in a dictionary format 'Smith': ['ROOT', 'nsubj']

## 2.Extract subtree of dependents given a token

Subtree is A sequence containing the token and all the token's syntactic descendants. Here I am extracting subtree of dependents from a token. In extractSubTreeDep the sentence is converted into tokens using nlp() and for each token in the sentence get its subtree using [tksb for tksb in token.subtree] and output of it is a list. which represents saved to the key, and its subtree, which is proposed as an ordered list and represents the value related.

## 3.Check if a given list of tokens (segment of a sentence) forms a subtree

In this part need to check if a given list is a subtree of a sentence or not check if the given list forms any subtree of the sentence. In checkSegofSenSubtree function accepts a sentence and the list or a sentence with subtree to compare.

After checking the probable subtree is sentence it its first converted to a list named sentenceList and then extracting the tokens from the sentence and then actual subtree is extracted.
If the given list is identical to any form of the subtree, then True is returned else false is returned.

## 4.Identify head of a span, given its tokens

spaCy's Matcher gives a Span-level information rather than Token-level because it allows a sequence of Tokens to be matched. In the same way that a Span can be composed of just 1 Token. In this part function headofspanis identifying the head from the given token. To get the head of a list of words, the list is joined, and a single sentence is created. Using nlp() the sentence is parsed as doc and span of the complete sentence is taken. Then next () to get just the first sentence as the expected input and at the end in-built function from spacy is call to find the root. The whole process is clubbed in a single line to make code more optimized.

```
next (nlp(' '.join(words)).sents).root
```

## 5.Extract sentence subject, object, direct and indirect object spans

**Direct Object**

A direct object will follow an action verb. Direct objects can be nouns, pronouns, phrases, or clauses. If you can identify the subject and verb in a sentence, then finding the direct object—if one exists—is easy. Just remember this simple formula:

Subject + Verb + what? or who? = Direct Object

**Indirect Object:**

When someone [or something] gets the direct object, that word is the indirect object. Look at these new versions of the sentences below:

Jim built his granddaughter a sandcastle on the beach.

**Jim** = subject; **built** = verb. Jim built what? **Sandcastle** = direct object. Who got the sandcastle? **Granddaughter** = indirect object.

Here function sub_dobj_iobj is extracting subject, direct and indirect object from spans.

In this function the sentence is accepted and parsed as doc using nlp () and then from doc token is iterated using for loop and each token is analysed whether it is Subject(nsubj), Direct Object(dobj) or Indirect Object (iobj).

Then for each of the subtrees the result is displayed as a list with the string of extraction about the word in the list.

```
nsubj, Stand for subject = :  ['Smith']
dobj, Stand for direct object = :  ['board']
iobj, Stand for indirect object = :  []
```