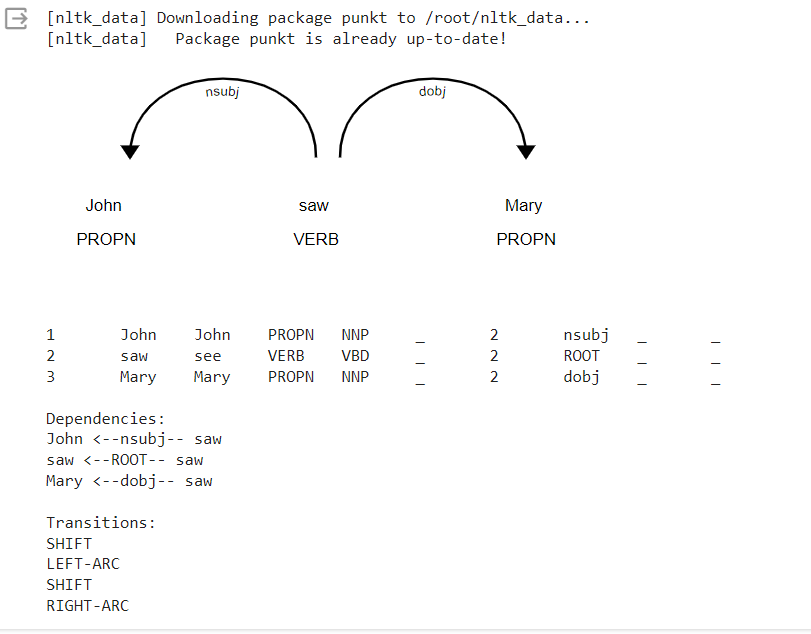
# **NLP Assignment 5**

# **112003066 – Om Khare – Division 1**

**Output:**



**Steps Involved:**

**1. Load Spacy Model and Input Sentence:**

- The code starts by loading the English spaCy model (`en\_core\_web\_sm`) and providing a sample sentence: "John saw Mary."

**2. Visualize Dependency Tree:**

- The sentence is processed using spaCy, and the dependency tree is visualized using the `displacy.render` function. This step shows the relationships between words in the sentence.

**3. Convert spaCy Doc to CoNLL Format:**

- The `spacy\_to\_conll` function is defined to convert the spaCy `Doc` object to CoNLL format, which is a tab-separated format commonly used for representing syntactic dependencies.

**4. Print CoNLL Formatted Output:**

- The CoNLL-formatted output is printed, displaying information about each token in the sentence, including its index, text, lemma, part of speech, tag, head index, dependency relation, etc.

**5. Define Transition Functions:**

- `SHIFT`, `LEFT\_ARC`, `RIGHT\_ARC`, and `REDUCE` are defined as transition labels. These labels represent the possible actions the transition-based parser can take.

**6. Define Oracle and Parser Functions:**

- The `oracle` function decides the next transition based on the current state of the stack, buffer, and dependencies.

- The `reduce` function checks if a reduction action is possible.

- The `transition\_based\_dependency\_parse` function performs the transition-based dependency parsing using the oracle function. It returns the parsed dependencies and the sequence of transitions.

**7. Example Usage:**

- The `transition\_based\_dependency\_parse` function is applied to the sample sentence, and the parsed dependencies and transitions are printed.

**8. Save Output to File:**

- The code then saves the dependencies and transitions in a readable format to a text file named "dependency\_parse\_output.txt."

**9. Download the File:**

- The text file is downloaded using the `files.download` function, allowing you to access the output file.

**Oracle Function:**

- The `oracle` function implements a simple strategy for deciding the next transition. It considers the top two elements on the stack and decides whether to perform a `LEFT\_ARC`, `RIGHT\_ARC`, `REDUCE`, or `SHIFT` action based on the relationships between these elements.

**Transition-Based Dependency Parsing:**

- The `transition\_based\_dependency\_parse` function uses a transition-based parsing approach to construct the dependency tree. It iteratively applies transitions until no valid transition is possible.

Overall, the code demonstrates a basic example of transition-based dependency parsing using a simple oracle strategy and outputs the results to a text file for further analysis.