

UNIVERSITÉ LIBRE DE BRUXELLES



INFO-Y404

NATURAL LANGUAGE PROCESSING

---

# Project : Dependency relations & Transition-based parser

---

Daoud YEMNI

20 avril 2018

# Table des matières

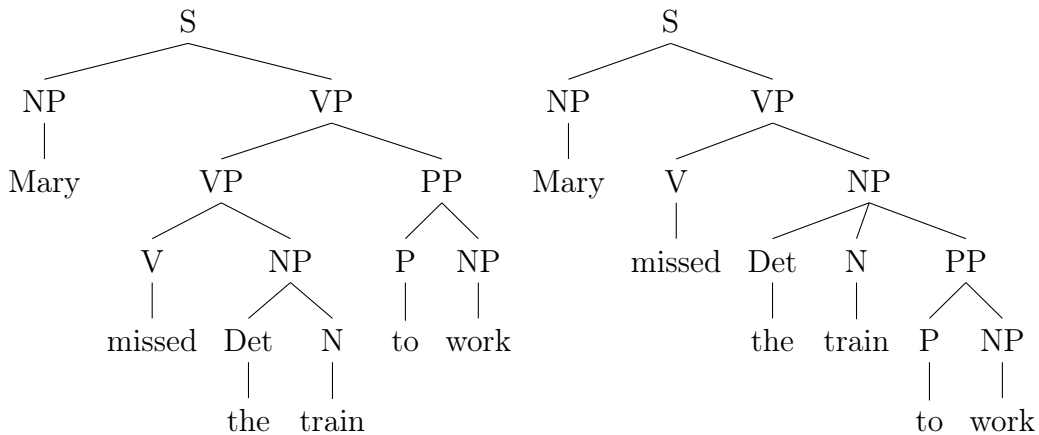
<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Part 1</b>	<b>2</b>
2.1	Dependency diagram . . . . .	3
2.2	Dependency tree . . . . .	4
<b>3</b>	<b>Part 2</b>	<b>4</b>
<b>4</b>	<b>Part 3</b>	<b>7</b>
4.1	Results . . . . .	7
4.2	Discussion . . . . .	8

# 1 Introduction

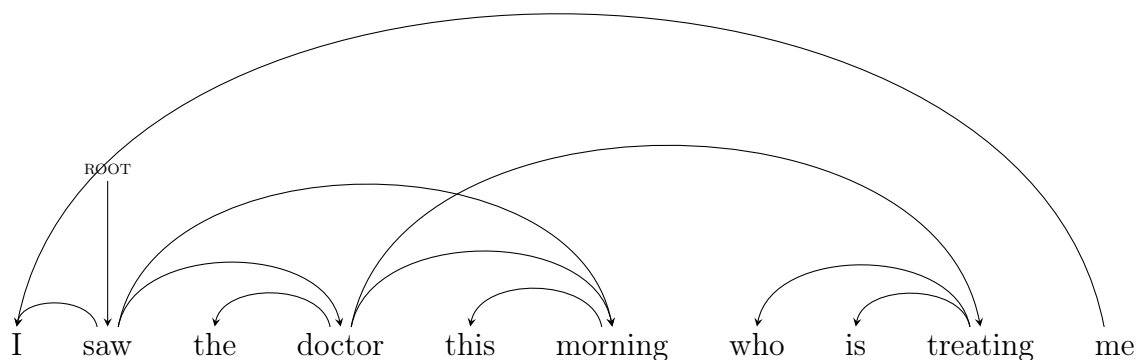
In this assignment, the ask is to create a transition-based parser and understand the dependency relations with 8 sentences. The first part of the project was to generate manually the CoNNL-U file on the 8 sentences. The CoNNL-U file is simply a format that give informations about each word of a sentence. The second part was to construct the training base for the Oracle, build the parser and using Oracle and read input file and generate 2 outputs : one is the CoNNL-U with some additional informations and other contains all traces of the configuration. A configuration is one step of the shift-reduce algorithm. Finally, in the last part, we needed to compare the CoNNL-U file created manually with the CoNNL-U file generated by the parser and compute the UAS, Unlabeled Attachment Accuracy. The Unlabeled Attachment Accuracy is simply compare the dependency edges between 2 dependency parse without looking the type of dependency relation.

## 2 Part 1

**there ambiguous sentences with multiple possible parses? If yes, what form does the ambiguity take?** Yes, there is some ambiguous sentences with multiple possible parses. The ambiguity has the form of the attachment ambiguity where the tree can be construct in multiple ways. For example, the sentence 2 "Mary missed her train to work" has 2 dependency trees :

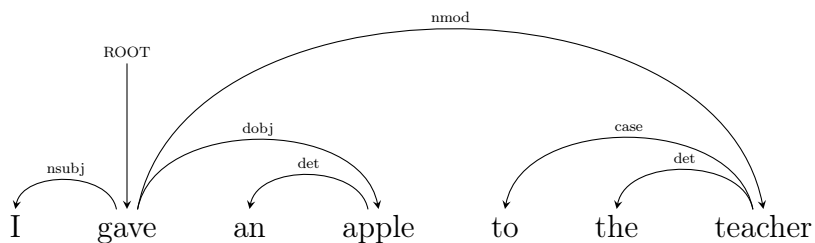


**Which sentence contains non-projective dependency relations ? Which relations exactly and why are they non-projective ?** The sentence number 6 contains a non-projective dependency relations.

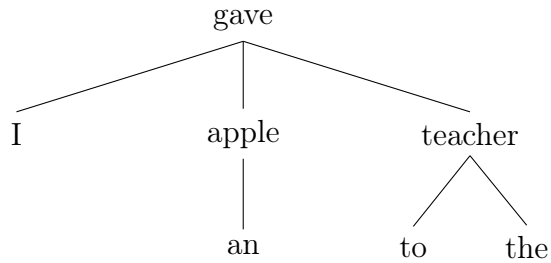


The relations ("saw", "morning") and ("doctor", "treating") are non-projective. They are non-projective because we have a collision in edge when we represent the words in a linear order.

## 2.1 Dependency diagram



## 2.2 Dependency tree



*I gave an apple to the teacher*

## 3 Part 2

First of all, I apply the shift reduce algorithm manually on the sentences 1-4 and I added 3 column : s1.t, s2.t and b1.t. I use those columns later to build the features template.

Stz	Word List	Action	Relation Added	s1.t	s2.t	b1.t
[root]	[I gave an apple to the teacher]	Shift		ROOT	null	PRON
[root, I]	[gave, an, apple, to, the, teacher]	Shift		PRON	root	VERB
[root, I, gave]	[an, apple, to, the, teacher]	LEFTARC	(I ← gave)	VERB	pron	DET
[root, gave]	[an, apple, to, the, teacher]	SHIFT		VERB	root	DET
[root, gave, an]	[apple, to, the, teacher]	SHIFT		DET	verb	NOUN
[root, gave, an, apple]	[to, the, teacher]	LEFTARC	(an ← apple)	NOUN	det	ADP
[root, gave, apple]	[to, the, teacher]	RIGHTARC	(gave → apple)	NOUN	verb	ADP
[root, gave, to]	[the, teacher]	SHIFT		ADP	verb	DET
[root, gave, to, the]	[teacher]	SHIFT		DET	adp	NOUN
[root, gave, to, the, teacher]	[]	LEFTARC	(the ← teacher)	NOUN	det	EMPTY
[root, gave, to, teacher]	[]	LEFTARC	(to ← teacher)	NOUN	adp	EMPTY
[root, gave, teacher]	[]	RIGHTARC	(gave → teacher)	NOUN	verb	EMPTY
[root, gave]	[]	RIGHTARC	(root → gave)	VERB	root	EMPTY
[root]	[]	Done		ROOT	null	EMPTY

FIGURE 1 – The shift reduce algorithm on the first sentence.

Sta	Word List	Action	Relation Added	s1.t	s2.t	b1.t
[root]	[Mary, missed, her, train, to, work]	SHIFT		ROOT	null	PROPN
[root, Mary]	[missed, her, train, to, work]	SHIFT		PROPN	root	VERB
[root, Mary, missed]	[her, train, to, work]	LEFTARC	(Mary ← missed)	VERB	pron	DET
[root, missed]	[her, train, to, work]	SHIFT		VERB	root	DET
[root, missed, her]	[train, to, work]	SHIFT		DET	verb	NOUN
[root, missed, her, train]	[to, work]	LEFTARC	(her ← train)	NOUN	det	ADP
[root, missed, train]	[to, work]	RIGHTARC	(missed → train)	NOUN	verb	ADP
[root, missed]	[to, work]	SHIFT		VERB	root	ADP
[root, missed, to]	[work]	SHIFT		ADP	verb	NOUN
[root, missed, to, work]	[]	LEFTARC	(to ← work)	NOUN	adp	EMPTY
[root, missed, work]	[]	RIGHTARC	(missed → work)	NOUN	verb	EMPTY
[root, missed]	[]	RIGHTARC	(root → missed)	VERB	root	EMPTY
[root]	[]	Done		ROOT	null	EMPTY

FIGURE 2 – The shift reduce algorithm on the second sentence.

Sta	Word List	Action	Relation Added	s1.t	s2.t	b1.t
[root]	[John, gave, the, teacher, a, very, heavy, book]	SHIFT		ROOT	null	PROPN
[root, John]	[gave, the, teacher, a, very, heavy, book]	SHIFT		PROPN	root	VERB
[root, John, gave]	[the, teacher, a, very, heavy, book]	LEFTARC	(John ← gave)	VERB	propn	DET
[root, gave]	[the, teacher, a, very, heavy, book]	SHIFT		VERB	root	DET
[root, gave, the]	[teacher, a, very, heavy, book]	SHIFT		DET	verb	NOUN
[root, gave, the, teacher]	[a, very, heavy, book]	LEFTARC	(the ← teacher)	NOUN	det	DET
[root, gave, teacher]	[a, very, heavy, book]	RIGHTARC	(gave → teacher)	noun	verb	DET
[root, gave]	[a, very, heavy, book]	SHIFT		verb	root	DET
[root, gave, a]	[very, heavy, book]	SHIFT		det	verb	ADV
[root, gave, a, very]	[heavy, book]	SHIFT		adv	det	ADJ
[root, gave, a, very, heavy]	[book]	LEFTARC	(very ← heavy)	adj	adv	NOUN
[root, gave, a, heavy]	[book]	SHIFT		adj	det	NOUN
[root, gave, a, heavy, book]	[]	LEFTARC	(heavy ← book)	noun	adj	EMPTY
[root, gave, a, book]	[]	LEFTARC	(a ← book)	noun	det	EMPTY
[root, gave, book]	[]	RIGHTARC	(gave → book)	noun	verb	EMPTY
[root, gave]	[]	RIGHTARC	(root → gave)	verb	root	EMPTY
[root]	[]	Done		root	null	EMPTY

FIGURE 3 – The shift reduce algorithm on the third sentence.

Sta	Word List	Action	Relation Added	s1.t	s2.t	b1.t
[root]	[the, sun, shines]	SHIFT		root	null	DET
[root, the]	[sun, shines]	SHIFT		det	root	NOUN
[root, the, sun]	[shines]	LEFTARC	(the ← sun)	noun	det	VERB
[root, sun]	[shines]	SHIFT		noun	root	VERB
[root, sun, shines]	[]	LEFTARC	(sun ← shines)	verb	noun	EMPTY
[root, shines]	[]	RIGHTARC	(root → shines)	verb	root	EMPTY
[root]	[]	Done		root	null	EMPTY

FIGURE 4 – The shift reduce algorithm on the fourth sentence.

After applying manually the shift reduce algorithm on the sentences 1-4. I decide to choose this template for each configuration :

$$< s1.t, op > \quad < s1.t \oplus s2.t, op > \quad < b1.t \oplus s2.t, op >$$

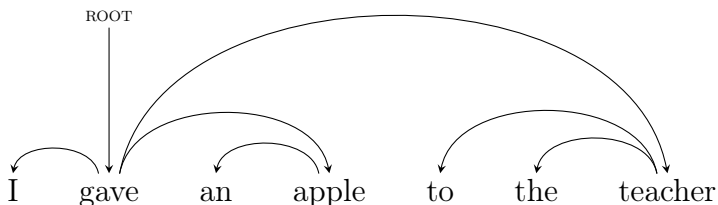
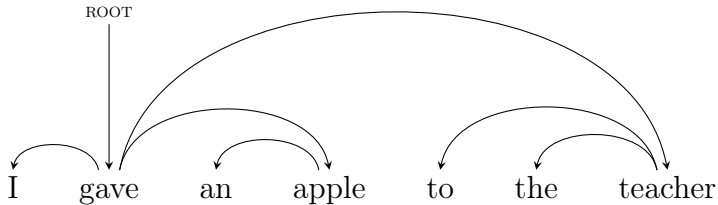
I added the one of double-word template because in the case where there is the verb and the root in the stack I have multiple possible actions. So the second double-word helps oracle to decide which action is needed. Here is an example of a features template for one configuration :

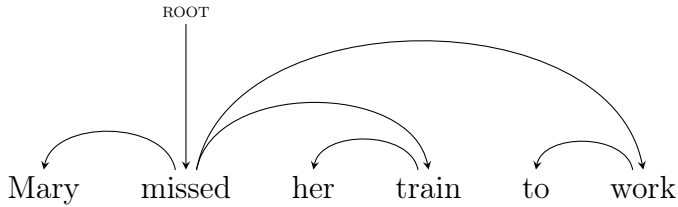
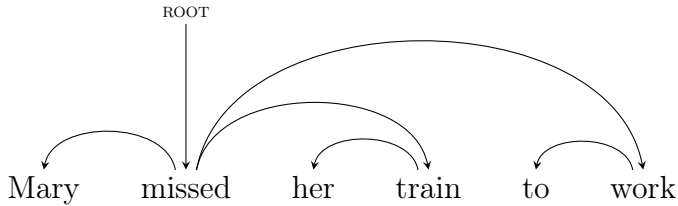
15    **ROOT, SHIFT**            **ROOTNULL, SHIFT**            **NULLPROP, SHIFT**

The process of my oracle is simple. Firstly we look into all double-word while we have multiple possible actions in a double-word. When we see all of double-word, we look into all single-word, in my case, we only look in  $\langle s1.t, op \rangle$ . I look in the template  $\langle s1.t \oplus s2.t, op \rangle$  the possible actions. If there is multiple possible actions, I will look into the template  $\langle b1.t \oplus s2.t, op \rangle$  and again if there is multiple action, I will finally look into the template  $\langle s1.t, op \rangle$ . In my case, the first double-word was sufficient for a majority expect one : the case where we have verb and root in the stack. For this only case, the process look into the second double-word.

## 4 Part 3

### 4.1 Results

Reference	
System	
UAS	7/7

Reference	
System	
UAS	6/6



Reference	
System	
UAS	8/8

Reference	
System	
UAS	3/3

## 4.2 Discussion

The system find the all of head-dependency relations for each sentence. The unlabeled attachment accuracy for each sentence is 100%. There exists multiple ways to produce the same result, this is called the *ambiguity*. But in my case, It seems Oracle learn very well and find always the correct transition

and re-produce exactly the same results as the manual one. I think it is totally normal that the parser didn't make mistakes. Oracle learned on the sentences 1-4 and we apply the parser on the same sentences. I think if we apply the Oracle on the other inputs, the parser may have difficulties or may make errors.