Natural Language Processing: Assignment 3

Shifei Chen

---

1 Dependency Annotation

Here are three typical errors I have made during my manual annotation.

1.1 `nmod` vs `compound`

> "If a Turkish employee quits, then the Turkish work councils come."

The ambiguation between a single `nmod` and a `compound` noun is confusing to me. In the phrase "work councils" I have annotated "work" as a nominal modifier to "councils" while the golden standard says they should be compounded. A compound word usually consists of nouns (or verbs, adjectives, etc.) being writted closely to each other. In fact in some Language like Swedish people usually write them together, without whitespaces. Meanwhile a nominal modifier is usually used to describe an attribute or genitive complement[1]. In this case, "work" can be argued to be the attribute of "councils" but since this is an idiom[2] so I agree with the golden standard.

1.2 `mark`

> "National reaction to the events in Kansas demonstrated how deeply divided the country had become."

Here I have marked the word "how" as a `mark` to the head word "become" rather than an `advmod` to the head word "deeply". In this sentence, "how deeply divided the country had become" is a complement clause, acting as the object to the verb "demonstrated". The word "become" is a `ccomp` to "demonstrated"[3]. The clause has indicated that its object is "divided", not "how", hence we should consider "how" as an adverbial modifer to the other adverb "deeply", even though sometimes "how" can be the mark word of a complement clause[4].

1.3 Subjects in Passive Voice

> "The RHS collected comments sent in by schoolchildren and teachers involved in the experiment."

I annotated "sent" to be (`acl`), the clause modifier of a noun, to "collected" and "schoolchildren" to be the nominal subject of "comments". The error in the former situation is obvious since "collected" is a verb[5]. I should have annotated "sent" to be the `acl` of "comments". The latter is also wrong. By the meaning of the sentence, it might be intriguing to annotate "schoolchildren" as a `nsubj` because those children are the one who sent comments. But there is an example from Universal Dependency explicly says "(`obl` is used) for the agent of a passive verb (with the optional subtype `obl:agent`):"[6]. We should mark "schoolchildren" as the `obl` to the verb "sent".

2 Dependency Parsing

Below are two examples of errors made by the parser in Lab 9.

> A witness told police that the victim had attacked the suspect in April.

In this sentence my parser annotated "police" as the `obj` of the verb "told", which should be an `iobj`. "Tell" can have a direct object like in the verb phrase "tell a lie" but here "police" is not the subject, the clause beginning with "that" is. This might be caused by our corpus containing too many "tell"s having direct objects. Also as my parser is a transition based parser, it always has a limited scope of words. It couldn't foresee the

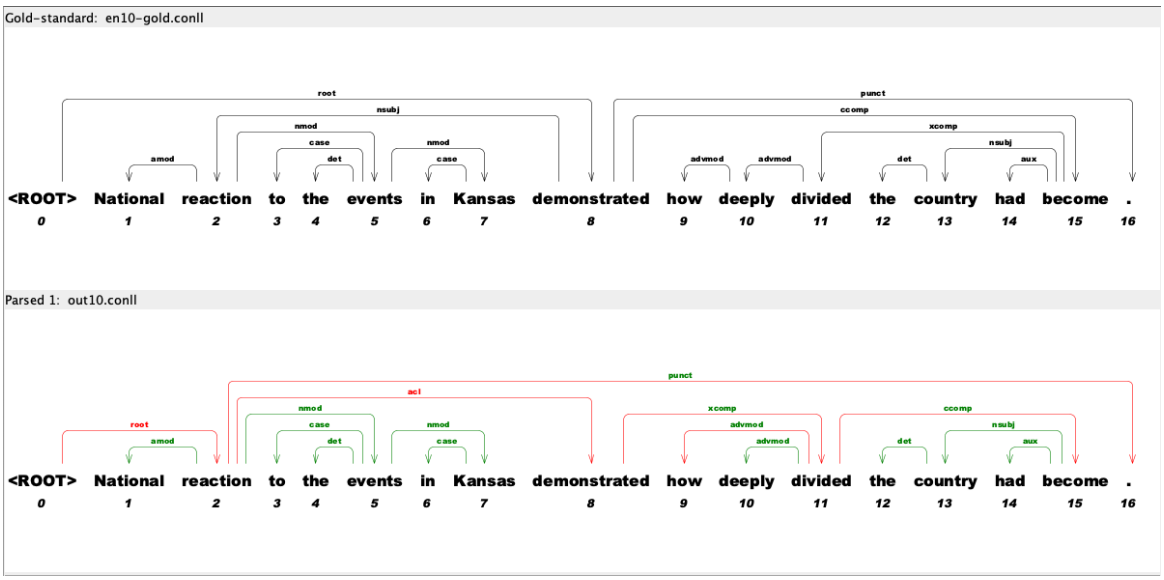


Figure 1: The Problem of Limited Scope

words behind ”police” therefore it prefered ”tell” to have a direct object. Nevertheless UD has a specific label `iobj` for this kind of indirect subject[7] and we should stick to that.

The problem of limited scope of words can also be demonstrated by another example in Figure 1.

My parser mistook the main structure of the sentence: it believed ”reaction” is the root therefore it continued to make further mistakes. I think this is the case of ”error propagation”. An arc-eager transition based parser was designed to do some greedy step to build arcs in order to save space in its stack, especially for sentences where the root word is located in the further back.

A transition based depnedency parser has three elements: a stack S, a buffer B for unprocessed words and a list A for arcs it has built. Starting with the root word in S, it scans the sentence from left to write and makes predictions learned from machine learning about whether to build a left arc or a right arc, to pop a word in the stack (reduce) or to push a word from the buffer to the stack (move). The termination point is when the parser empties its buffer and at that time A is the complete list of parsed arcs.

We shall look at this sentence as an example to show how the parser works.

$$[\texttt{ROOT}]_s[\texttt{He worked for the BBC for a decade .}]_B \tag{1}$$

We start with root being our W_0.

$$[\texttt{ROOT He}]_s[\texttt{worked for the BBC for a decade .}]_B \tag{2}$$

Shift ”He” into S

$$[\texttt{ROOT}]_s[\texttt{worked for the BBC for a decade .}]_B \tag{3}$$

Build a left arc from ”worked” to ”He”, label it `nsubj`.

$$[\texttt{ROOT worked}]_s[\texttt{for the BBC for a decade .}]_B \tag{4}$$

Build a right arc from root to ”worked”, label it `ROOT`.

$$[\texttt{ROOT worked for the}]_s[\texttt{the BBC for a decade .}]_B \tag{5}$$

Shift ”for” and the.

$$[\texttt{ROOT worked for}]_s[\texttt{BBC for a decade .}]_B \tag{6}$$

Build a left arc from ”BBC” to ”the”, label it `det`.

$$[\texttt{ROOT worked}]_s[\texttt{BBC for a decade .}]_B \tag{7}$$

Build a left arc from ”BBC” to ”for”, label it `case`.

$$[\texttt{ROOT worked BBC}]_s[\texttt{for a decade .}]_B \tag{8}$$

Build a right arc from ”worked” to ”BBC”, label it `obl`.

$$[\texttt{ROOT worked BBC for a}]_s[\texttt{decade .}]_B \tag{9}$$

Shift ”for” and ”a”.

$$[\texttt{ROOT worked BBC for}]_s[\texttt{decade .}]_B \tag{10}$$

Build a left arc from ”decade” to ”a”, label it `det`.

$$[\texttt{ROOT worked BBC}]_s[\texttt{decade .}]_B \tag{11}$$

Build a left arc from ”decade” to ”for”, label it `case`.

$$[\texttt{ROOT worked}]_s[\texttt{decade .}]_B \tag{12}$$

Reduce ”BBC”.

$$[\texttt{ROOT worked decade}]_s[\texttt{.}]_B \tag{13}$$

Build a right arc from ”worked” to ”decade”, label it `obl`.

$$[\texttt{ROOT worked}]_s[\texttt{.}]_B \tag{14}$$

Reduce ”decade”.

$$[\texttt{ROOT worked .}]_s[]_B \tag{15}$$

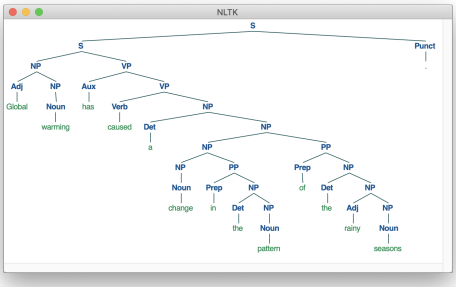
Build a right arc from ”worked” to ”.”, label it `punc`. The empty buffer B marks our end of parsing.

3 Context-Free Grammar

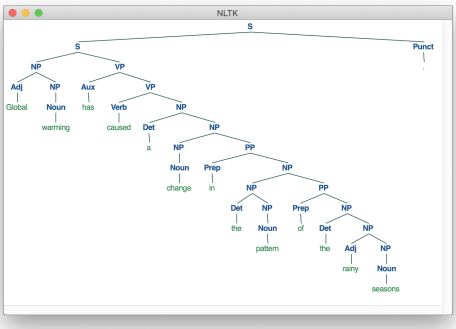
My grammar goes as the following one.

```
# Grammar
S -> S Punct
S -> NP VP
NP -> NP PP | Det NP | Adj NP
NP -> Pronoun | Pronoun Pronoun | Noun | Noun Cconj Noun
VP -> Aux VP | Adv VP
VP -> Verb NP | Verb SBAR | Verb PP | Verb PP PP
PP -> Prep NP
SBAR -> Sconj S
# Lexicon
Noun -> 'decade' | 'experience' | 'warming' | 'change' | 'pattern' |
  'seasons' | 'part' | 'scheme' | 'money' | 'sponsorship' | 'advertising'
Verb -> 'worked' | 'spoke' | 'caused' | 'wonder' | 'played' | 'makes'
Adj -> 'Global' | 'rainy'
Aux -> 'has'
Adv -> 'also'
Sconj -> 'whether'
Cconj -> 'and'
Pronoun -> 'BBC' | 'He' | 'She' | 'CNN' | 'Style' | 'I' | 'Davis' | 'Cup'
Det -> 'the' | 'a' | 'The'
Prep -> 'for' | 'to' | 'about' | 'in' | 'of' | 'through'
Punct -> '.'
```

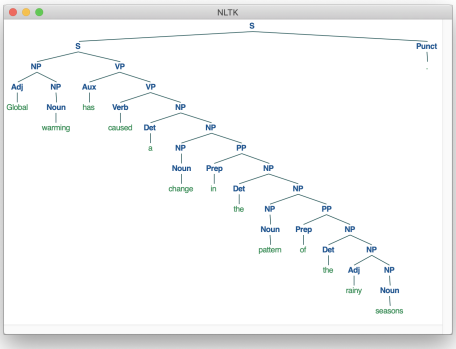
What I have extended are the rules for noun pharses and verb phrases. A noun phrase can consist of several nouns, pronouns or nouns connected by a coordinating conjunction. In additional it can also consist of a noun followed by one or several prepositional phrases, e.g. ”a change in the pattern” and ”the fire in December”. For verbs they can have nouns, one/several prepositional phrase, or even a subjordinate clause as their object. There are also cases that a verb follows an auxiliary verb like ”has caused”, or cases that it follows an adverb,



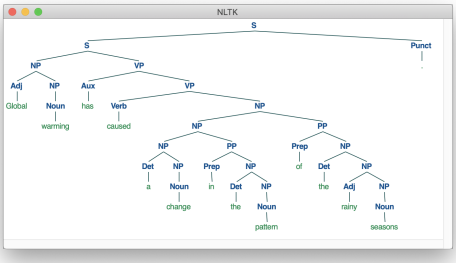
(a)



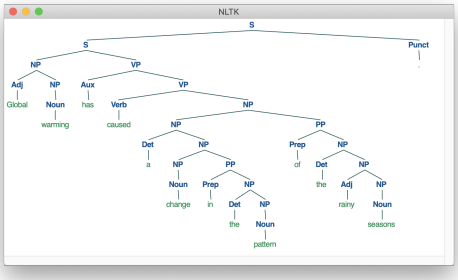
(b)



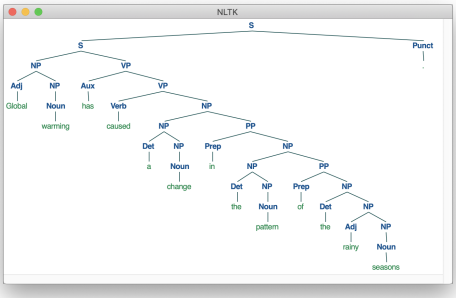
(c)



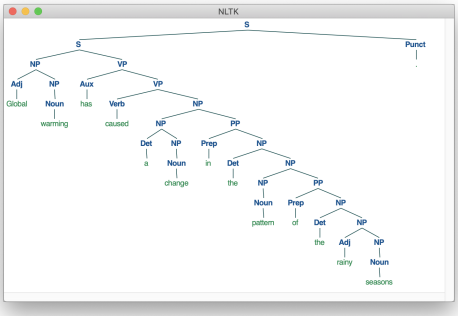
(d)



(e)



(f)



(g)

Figure 2: Different Trees Generated by My Grammar

like "also wonder", even thought some may argue that the adverb "also" in this case should not be a part of the verb phrase because it modifies the whole sentence, not only the verb.

My grammar has generated 7 different trees for the sentence "Global warming has caused a change in the pattern of the rainy seasons." The left part of the sentence "Global warming has caused" and the punctuation are the same in all trees. They differ in parsing the big `NP` "a change in the pattern of the rainy seasons". Only one of them(Figure 2c) picked "change" to be the head word of "pattern" and "pattern" to be the head word of "seasons", which is what the golden standard did as well. I believe the variety, or ambiguaty is caused by my recursive definition-`NP` $\rightarrow$ `NP PP` and `PP` $\rightarrow$ `Prep NP`. If we could assign probabilities to each grammar rule then I believe we can disambigute this sentence.

In the lecture we have learned CYK algorithm to parse sentences. In order to apply it, first we need to convert our CFG to the one in Chomsky normal form[8, p. 436]. After that we draw the upper-right portion of a $(n+1)*(n+1)$ matrix for a n words sentence, each of its cell $[i, j]$ will hold a set of non-terminals representing all of the consitituents spanning from position i through j. We first fill out all the cells that are located on the diagonal line by looking for the input in the CNF grammer. The results are usually their corrsponding pos tags. Then we move from left to right in the table and for each column we start at the second but last cell at the bottom all the way to the top. For cell $[i, j]$ we look at cell $[i, x]$ on the left in the same row and the cell $[x, j]$ below in the column to search for the first and the second non-terminals which can be the right-hand part of a rule in our CNF grammar (order matters). We write down the left hand side non-terminal in the cell $[i, j]$. If there are several rules exist, we need to write down all of the possible non-terminals. Then we continue filling up the table until cell $[0, n]$. Figure 3 below is my final table for parsing the sentence "He worked for the BBC for a decade."

References

[1] *nmod*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/nmod.html`.

[2] *Works council - Wikipedia*. [Online; accessed 5. Dec. 2018]. 2018. URL: `https://en.wikipedia.org/wiki/Works_council`.

[3] *ccomp*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/ccomp.html`.

[4] *mark*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/mark.html`.

[5] *acl*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/acl.html`.

[6] *obl*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/obl.html`.

[7] *iobj*. [Online; accessed 5. Dec. 2018]. 2017. URL: `http://universaldependencies.org/u/dep/iobj.html`.

[8] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, 2nd Edition*. 2nd. Prentice Hall, May 2008. ISBN: 9780131873216. URL: `http://amazon.com/o/ASIN/0131873210/`.

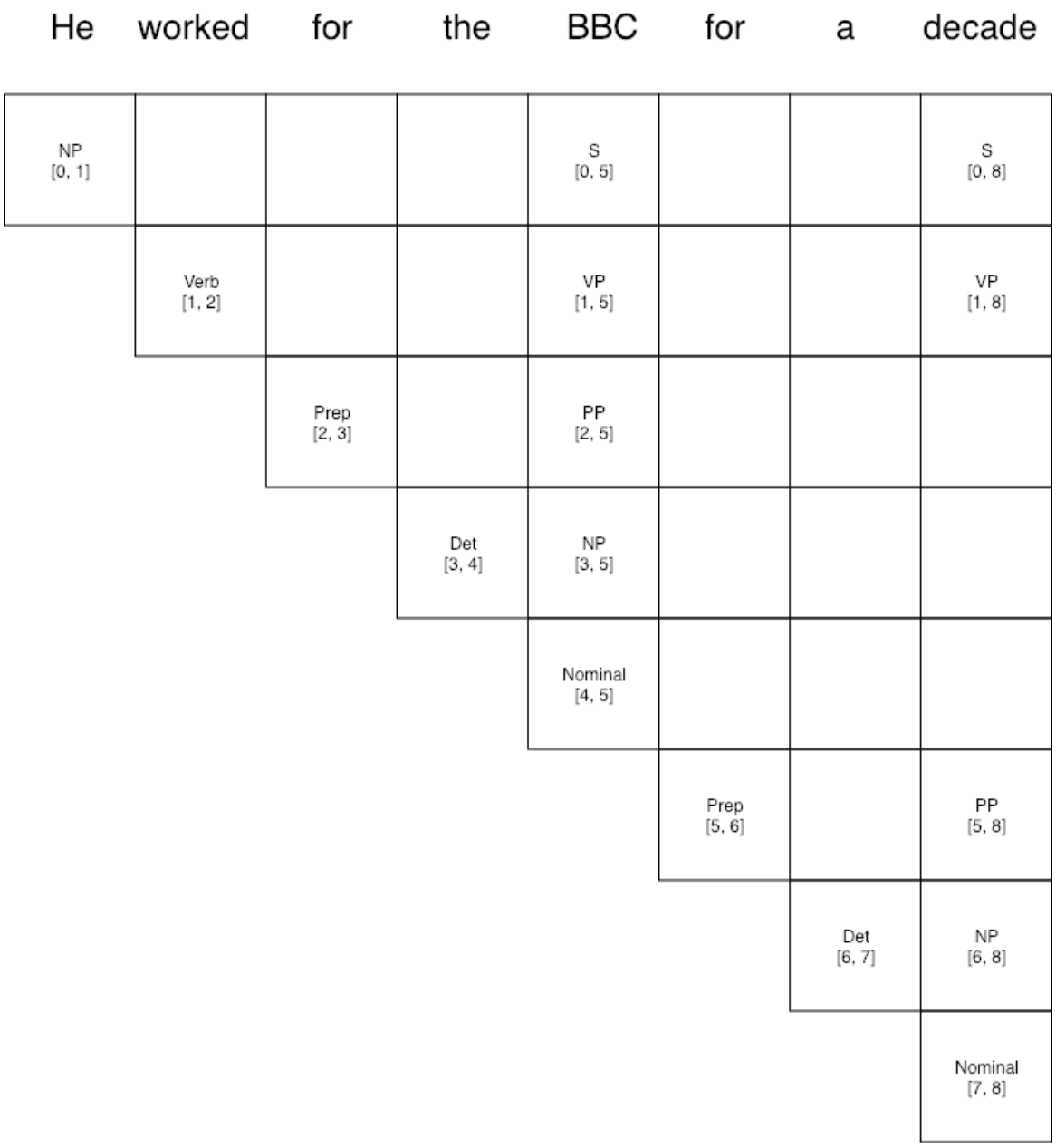


Figure 3: Parsing table by using the CYK algorithm