LG 467 Computers in Linguistics

[1-2021] Topic 6: Parsing

Sakol Suethanapornkul

Previously...

Context-Free Grammar (CFG)

- A formal system for modeling constituent structure
- A set of (de)composition rules over a set of symbols

"G" is defined by four parameters:

- N Set of **non-terminal** symbols
- Σ Set of **terminal** symbols (not in N)
- R Set of rules, each in the form $A \rightarrow \beta$, where $A \in N$, $\beta \in (\Sigma \cup N)^*$
- S Designated start symbol

Previously...

Penn Treebank Project (PTB): syntactically annotated corpus with constituent analyses

Basic idea:

- Parse a sentence using a labeled bracketing structure
- POS tags are integrated into a tree
- Not necessarily binary (e.g., **NP the black cat**)

Previously...

<u>Example</u>: The dog bit the cat.

```
(S
    (NP (DT The) (NN dog) )                    #subject above VP
    (VP (VVD bit)
        (NP (DT the) (JJ black) (NN cat) )     #object above VP
    )
.)
```

Previously...

PTB trees contain a lot of useful information:

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Source: Figure 12.7 in Jurafsky & Martin [chapter 12]

Simpler treebanking?

For many applications, we just want to know:

- which words belong together in a phrase
- what the head is
- what the basic hierarchy is

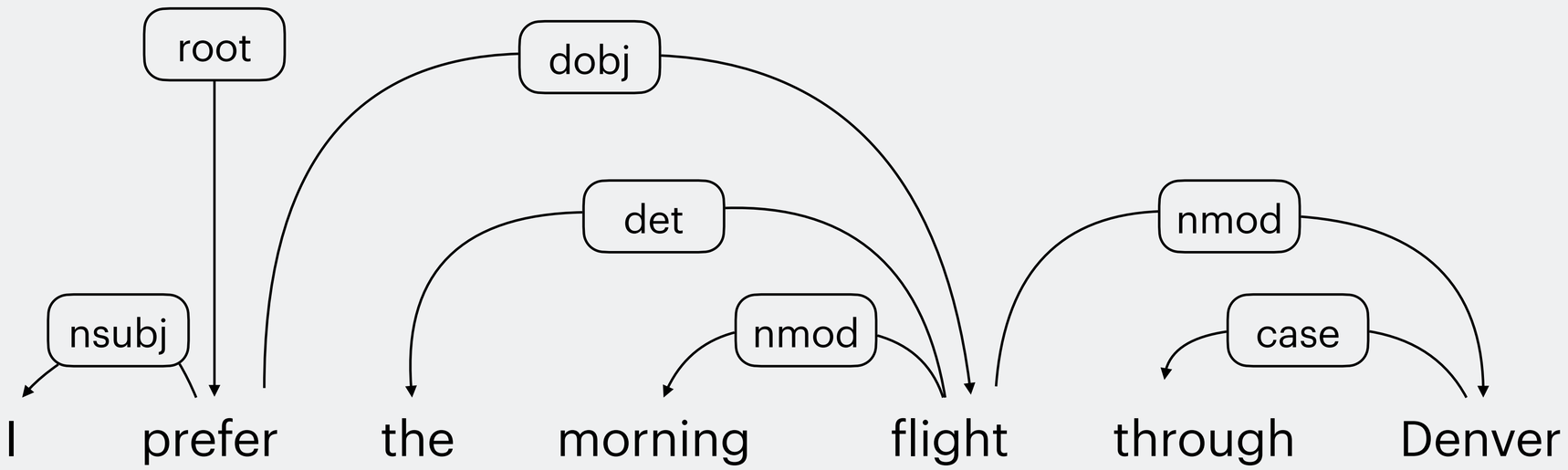
In other words, we want a simpler syntax annotation (that can also work with other languages)

So, let's talk about **dependency grammar**

Dependency Grammar

Dependency grammar: Overview

In dependency grammar, syntactic structure is described in terms of relations between words of a sentence (**heads** and **dependents**)



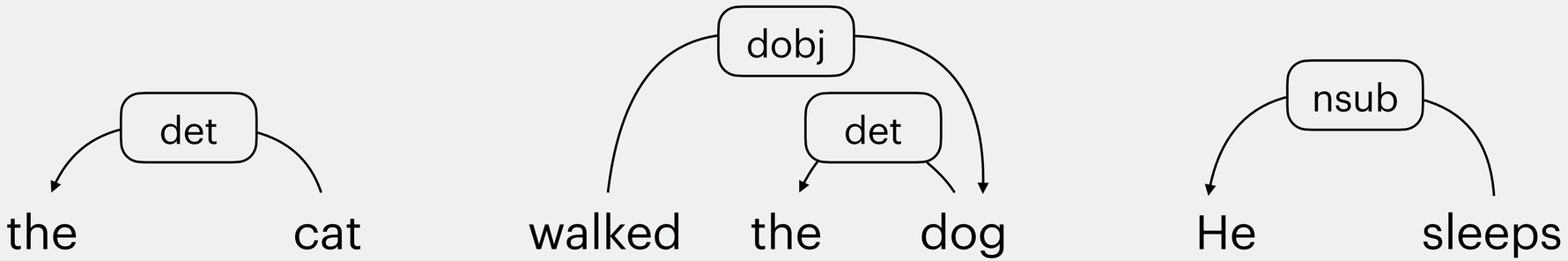
Dependency grammar: Overview

Heads: central organizing words of a larger constituent

- the **cat**, **walked** the dog, He **sleeps** like a baby

Dependents: remaining words in the constituent

- **the** cat, walked the **dog**, **He** sleeps like a **baby**



Dependency grammar: Overview

Key points:

- Every word depends on exactly one other (except the 'root')
- Arguments "depend" on the predicate (= root)
- Adjuncts depend on the head they modify
- No empty categories
- No non-terminal categories–just words (what you see is what you get)

Dependency relations

Universal Dependencies (UD) project → dependency relations

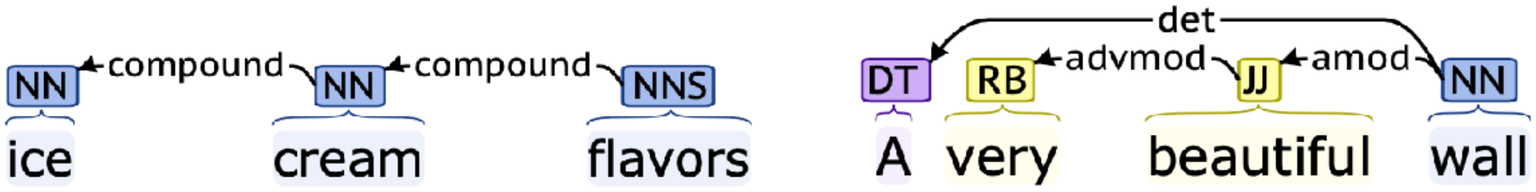
- Two main sets:
  - clausal relations (describing syntactic roles re: a predicate (verbs))
  - modifier relations (categorizing ways in which words can modify their heads)

Lesson 1: The NP domain

The head noun is the source of all edges:

- det — determiner — **the** car
- amod — adjective modifier — **good** news
- advmod — adverb modifier — **very** tough decision
- compound — noun compound — **phone** book

NP ≈ a chain of dependencies

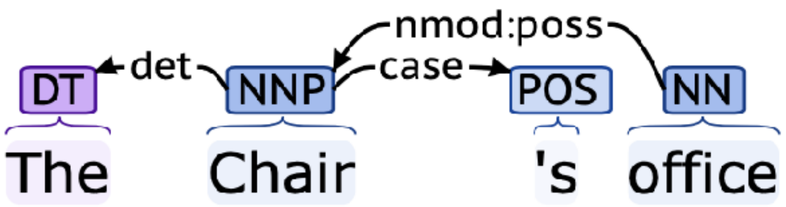


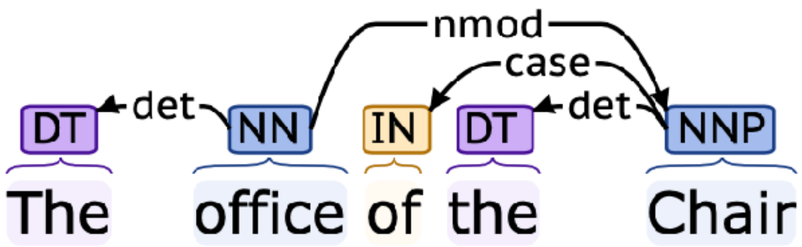
Source: corenlp.run

Lesson 1: The NP domain

Possessives need up to two labels:

- nmod:poss possessive marker The **chair**'s office
- case case-marking element The chair**'s** office





- nmod nom. dependents of n. Flight through **Denver**

Source: corenlp.run

Lesson 1: Exercise

Come up with a dependency analysis of the following NP:

The very expensive car of your friend

Labels: *det, amod, advmod, case, nmod, nmod:poss, root*

Consider these:

- Where's the root?
- What's a determiner (det)?

Lesson 1: Exercise

Come up with a dependency analysis of the following NP:

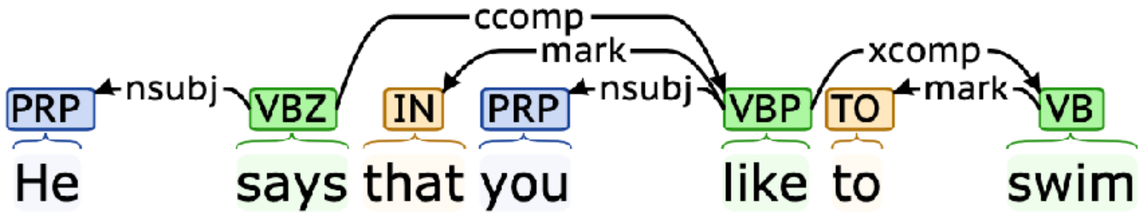
The very expensive car of your friend

Answer:

Lesson 2: The verb and its arguments

We are in for quite a challenging ride!

• nsubj	nominal subject	**He** cares...
• obj	direct object	She loves **you**.
• iobj	indirect object	.. send **me** a letter.
• compound:prt	clausal complement	They shut **down** the plant
• ccomp	clausal complement	He says you **like** to swim



• compound: svc serial verb compound

Source: corenlp.run

Lesson 2: Exercise

Come up with a dependency analysis of the following sentence:

They owe me a nice dinner.

NOTE: Ignore punctuation for this exercise.

Lesson 2: Exercise

Come up with a dependency analysis of the following sentence:

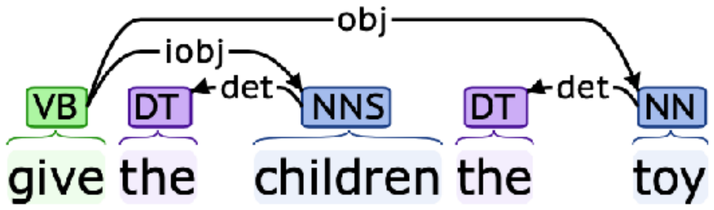
They owe me a nice dinner.

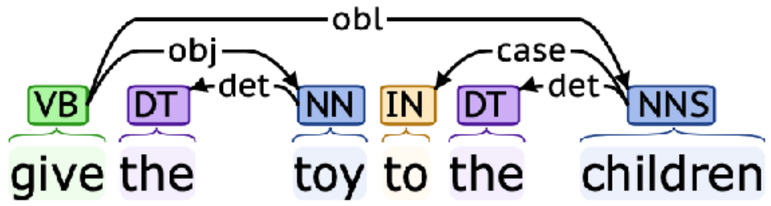
Answer:

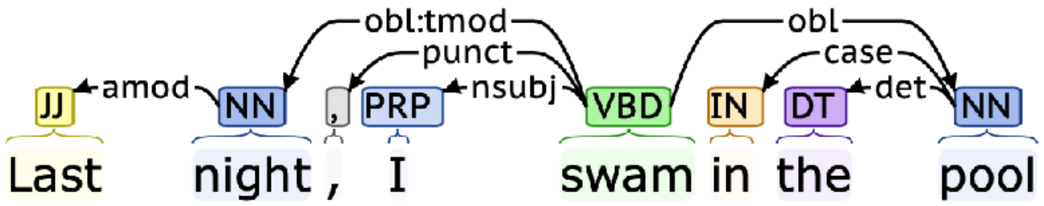
Lesson 2: The verb and its arguments

We are in for quite a challenging ride!

- obl oblique nominal He sent a letter to **me**





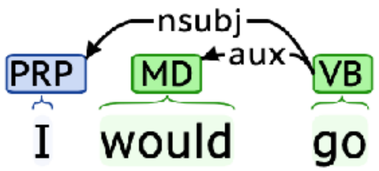


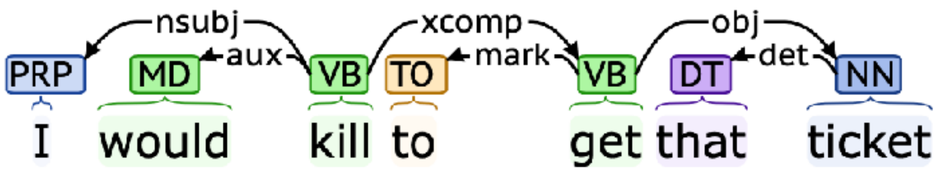
Source: corenlp.run

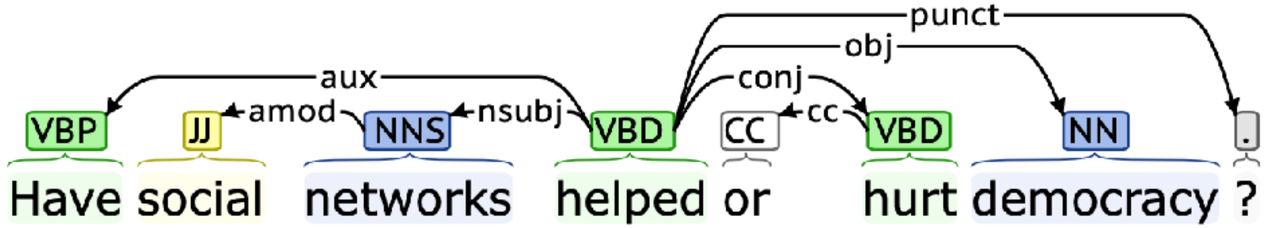
Lesson 3: Auxiliaries and coordination

Let's extend what we know to deal with coordination

- aux — auxiliary (dependent of lexical verbs; incl. modals) — I **would** go
- cc — coordinating conjunction — come **and** go
- conj — a conjunct — come and **go**





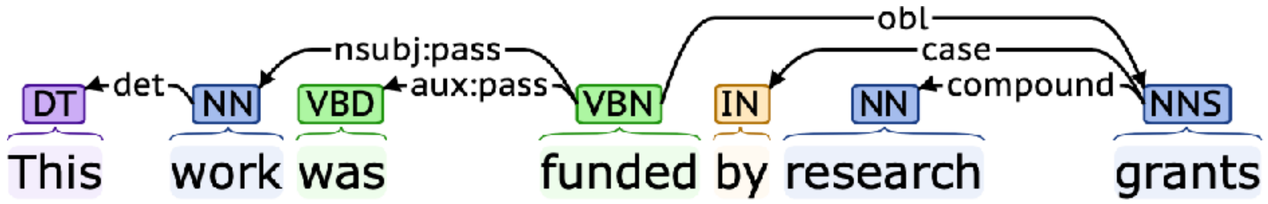


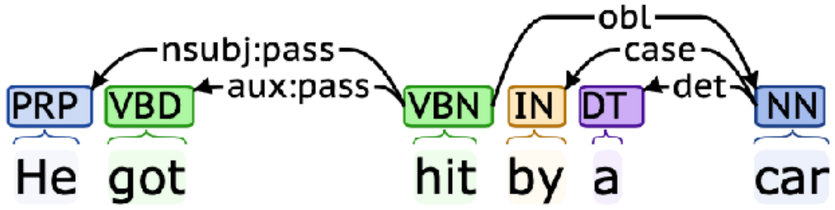
Source: corenlp.run

Lesson 4: Passive

Passives get special labels:

- nsubj:pass passive subject I was duped
- aux:pass passive auxiliary I **was** duped



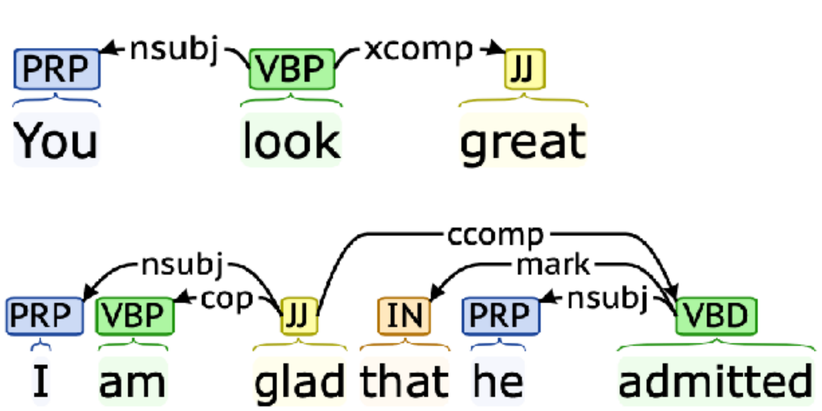


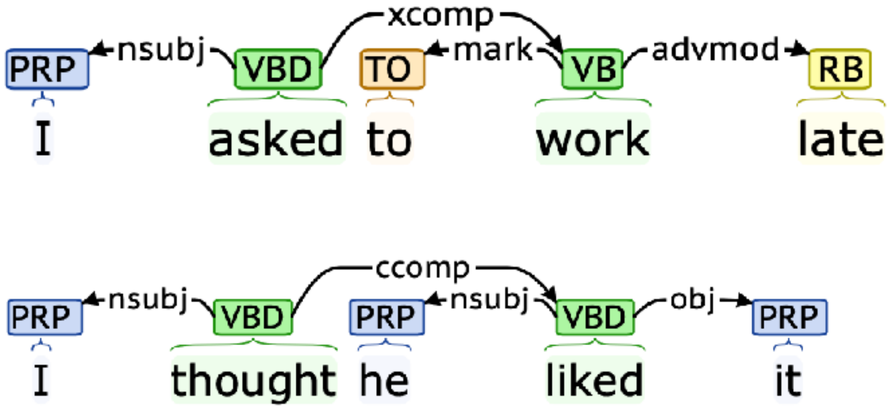
Source: corenlp.run

Lesson 5: Clauses

There are quite a few labels for clauses:

- xcomp open clausal complement I wanted to **work** for him
- ccomp complement clause I know that you **went**





Source: corenlp.run

Lesson 4 & 5: Exercise

Come up with a dependency analysis of the following sentence:

I did not know that he was selected.

Lesson 4 & 5: Exercise

Come up with a dependency analysis of the following sentence:

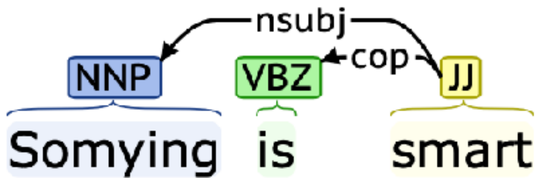
I did not know that he was selected.

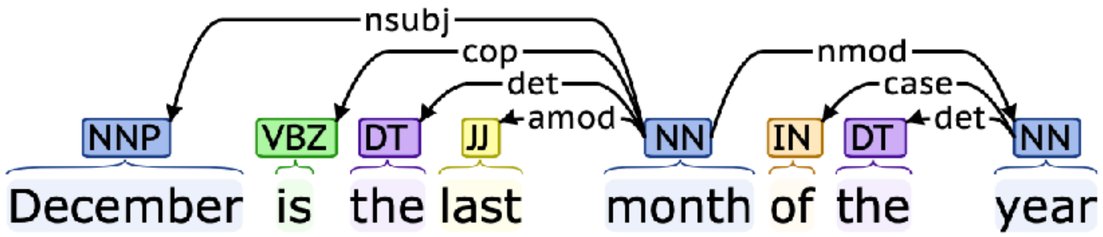
Answer:

Lesson 6: Copula sentences

Copula sentences treat the (nominal) predicate as root

- cop copula She **is** smart





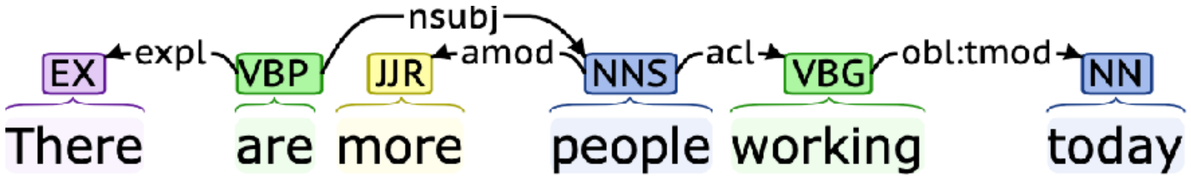
NOTE: The subject is still *nsubj* to the predicate

Source: corenlp.run

Lesson 7: "There is" and adnominal clauses

There is/are receives a special label, as well as participles

- expl copula **There** are three issues
- acl clausal modifier of noun A man **walking** the dog is...

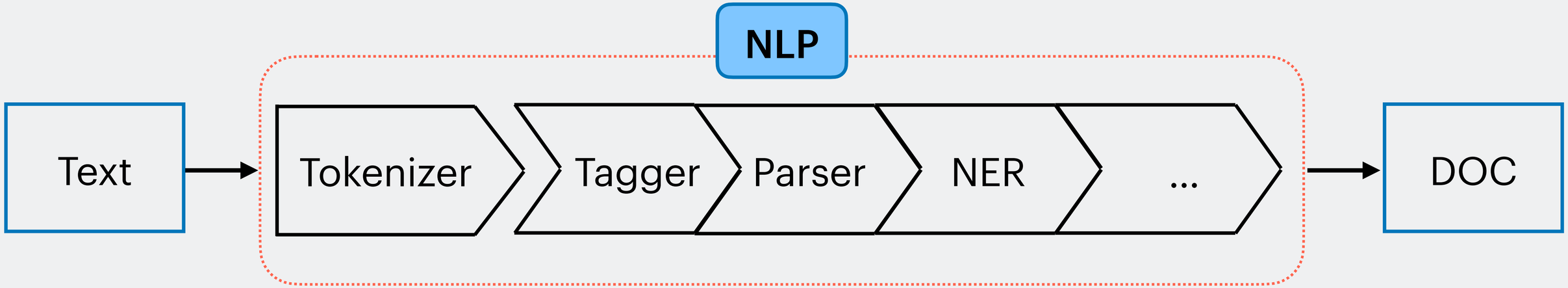


Source: corenlp.run

SpaCy

Processing pipeline in SpaCy

SpaCy applies several processing steps to produce a `Doc` object



Adapted from: Spacy's website

Processing pipeline in SpaCy

A SpaCy pipeline can be initialized with the following code:

Code 10.1

```
# Import SpaCy
import spacy
from spacy import displacy

# Load the model & initialize a pipeline
nlp = spacy.load("en_core_web_sm")

# Apply our pipeline on a text
doc = nlp("December is the last month of the year")
```

Processing pipeline in SpaCy

Each pipeline component has a well-defined task:

Name	Description	Creates
tagger	Part-of-speech tagger	Token.tag, Token.pos
parser	Dependency parser	Token.dep, Token.head, Doc.sents, Doc.noun_chunks
ner	Named entity recognizer	Doc.ents, Token.ent_iob, Token.ent_type

POS tagging in SpaCy

Recall that we can obtain POS tags by calling:

Code 10.2

```
import pandas as pd

# Review: Obtaining POS tags
for tok in doc:
    print(tok.i, tok.text, tok.pos_, tok.tag_)

# Write our analysis to a csv file
df = pd.DataFrame({'Tokens':  [tok.text for tok in doc],
                   ‘UD_tags': [tok.pos_ for tok in doc],
                   'PTB_tags':[tok.tag_ for tok in doc]
                  })

df.to_csv('file.csv')
```

Dependency parse in SpaCy

The parser creates labels that we can use:

Code 10.3

```
for tok in doc:
    print(tok.i, tok.text, tok.dep_, tok.head)

#0 December nsubj is
#1 is ROOT is
#2 the det month
#3 last amod month
#4 month attr is
#5 of prep month
#6 the det year
#7 year pobj of

spacy.explain('pobj')
```

Dependency parse in SpaCy

Let's use a different sentence:

```
doc1 = nlp("He was hit by a bus yesterday")

for tok in doc1:
    print(tok.i, tok.text, tok.dep_, tok.head)

displacy.serve(doc1, style = 'dep')
```

Code 10.4

See this link on how to save an image file

Dependency parse in SpaCy

Not everything is perfect; your own analysis can also be wrong!

- Things are more right than wrong

Our plan for the final week…

- We will wrap up with what we haven't covered (which is a lot)
- We will also discuss biases in NLP
- Reading
  - Hovy & Prabhumoye (2021). Five sources of biases