Lecture 22:
More on Syntax & Parsing, WordNet

Ling 1330/2330 Intro to Computational Linguistics

Na-Rae Han, 11/14/2023

Overview

- Linguistic exploration of trees
- Probabilistic CFG
- Dependency grammar
  - J& M Ch.18 Dependency parsing
  - https://www.nltk.org/book/ch08.html#dependencies-and-dependency-grammar
- Computational semantics
  - WordNet

Linguistic exploration, probabilistic CFG

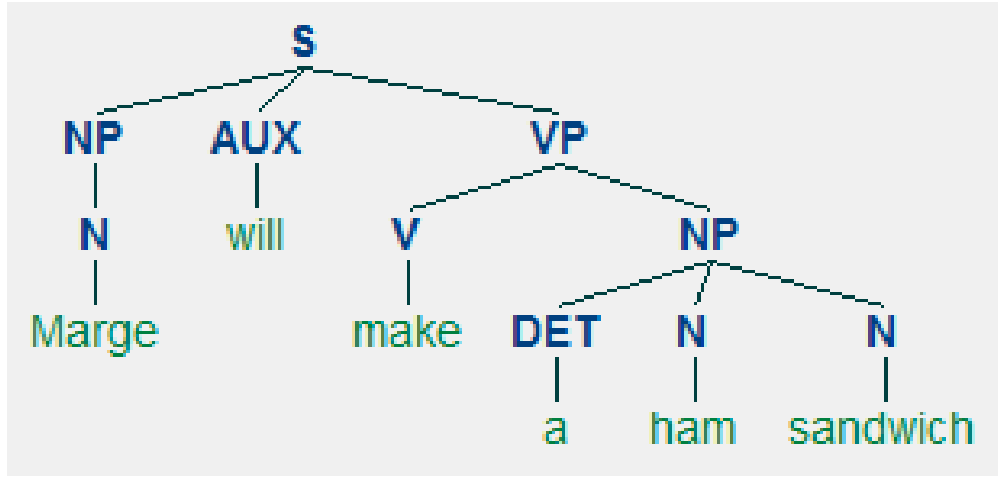
- HW8 review, probabilistic CFG
  - https://sites.pitt.edu/~naraehan/ling1330/lecture22.html

- Linguistic exploration of trees
  - https://sites.pitt.edu/~naraehan/ling1330/lecture21.html

Context-free grammar

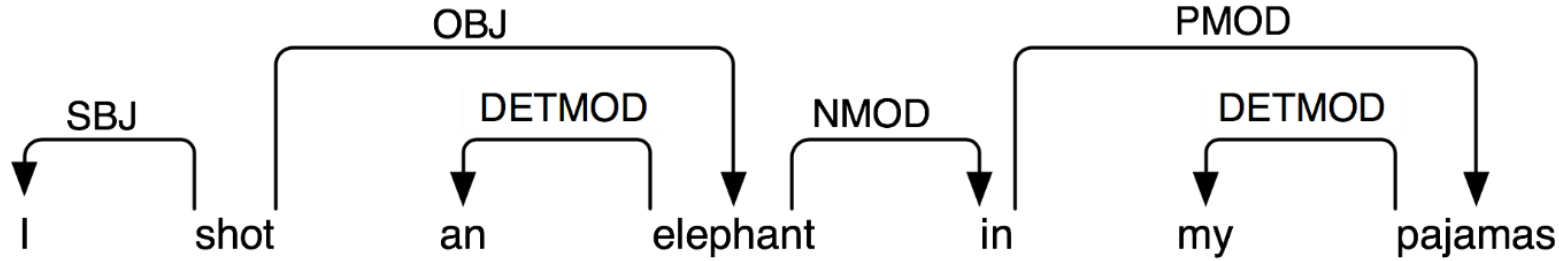
- Phrase-structure grammar is based upon constituency.
- Each local constituent can be expressed through **context-free grammar**.

```
S -> NP AUX VP
NP -> N
VP -> V NP
NP -> DET N N
N -> 'Marge'
Aux -> 'will'
V -> 'make'
DET -> 'a'
N -> 'ham' | 'sandwich'
```



A paradigm shift: dependency grammar

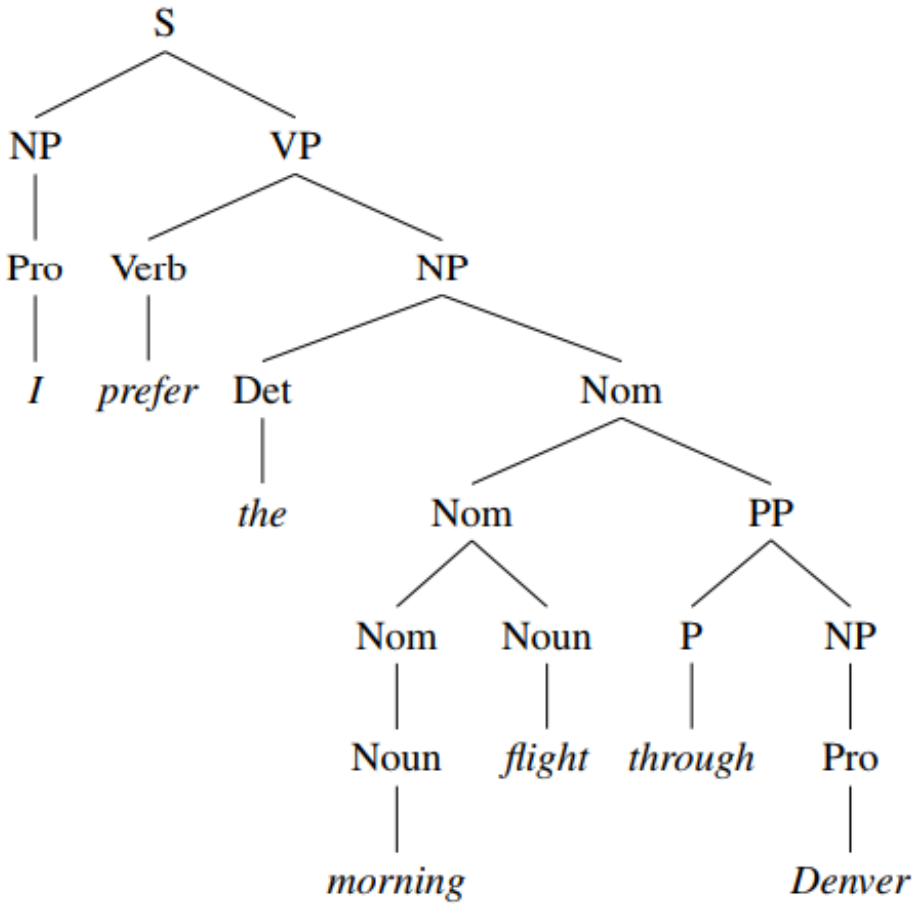
- **Phrase structure grammar** is all about **constituents**: phrasal units that words combine into.
- **Dependency grammar**, on the other hand, focuses on how words *relate* to other words: dependency relation between the headword and its dependents.

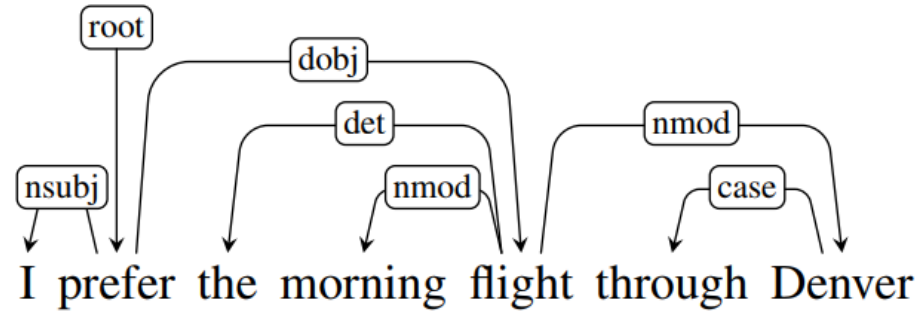


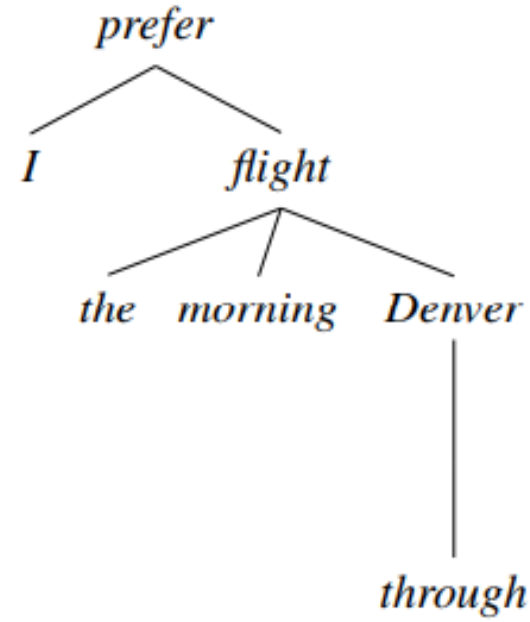
- A sentence's head (= "root") is the main **verb**.
- Dependency grammars can be used to directly express **grammatical functions** as a type of dependency.
- NLTK's section:
  - https://www.nltk.org/book/ch08.html#dependencies-and-dependency-grammar

A comparison

Constituency grammar vs. Dependency grammar







A constituent tree can be algorithmically converted to a dependency tree, via **head rules**.

P is the head of PP, V is the head of VP.

Universal dependencies

- Dependency grammar and parsing have become increasingly popular.
- Dependency grammar is thought to be more suited to languages with flexible word order.

🡐 Could it be a better candidate for a truly universal grammar formalism?

🡐 Linguistic theory aside, does it offer an engineering-side advantage?

- **Universal Dependencies** working group
  - https://universaldependencies.org/introduction.html
  - A wide variety of languages represented!
    - https://universaldependencies.org/

Dependency annotation: example

- https://raw.githubusercontent.com/UniversalDependencies/UD_English-PUD/master/en_pud-ud-test.conllu

```
# sent_id = n01001013
# text = For those who follow social media transitions on Capitol Hill, this will be a little different.
1	For	for	ADP	IN	_	2	case	2:case	_
2	those	those	PRON	DT	Number=Plur|PronType=Dem	17	obl	4:nsubj|17:obl:for	_
3	who	who	PRON	WP	PronType=Rel	4	nsubj	2:ref	_
4	follow	follow	VERB	VBP	Mood=Ind|Tense=Pres|VerbForm=Fin	2	acl:relcl	2:acl:relcl	_
5	social	social	ADJ	JJ	Degree=Pos	6	amod	6:amod	_
6	media	media	NOUN	NN	Number=Sing	7	compound	7:compound	_
7	transitions	transition	NOUN	NNS	Number=Plur	4	obj	4:obj	_
8	on	on	ADP	IN	_	10	case	10:case	_
9	Capitol	Capitol	PROPN	NNP	Number=Sing	10	compound	10:compound	_
10	Hill	Hill	PROPN	NNP	Number=Sing	7	nmod	7:nmod:on	SpaceAfter=No
11	,	,	PUNCT	,	_	17	punct	17:punct	_
12	this	this	PRON	DT	Number=Sing|PronType=Dem	17	nsubj	17:nsubj	_
13	will	will	AUX	MD	VerbForm=Fin	17	aux	17:aux	_
14	be	be	AUX	VB	VerbForm=Inf	17	cop	17:cop	_
15	a	a	DET	DT	Definite=Ind|PronType=Art	16	det	16:det	_
16	little	little	ADJ	JJ	Degree=Pos	17	obl:npmod	17:obl:npmod	_
17	different	different	ADJ	JJ	Degree=Pos	0	root	0:root	SpaceAfter=No
18	.	.	PUNCT	.	_	17	punct	17:punct	_
```

Dependency annotation: example

- https://raw.githubusercontent.com/UniversalDependencies/UD_Spanish-PUD/master/es_pud-ud-test.conllu

```
# sent_id = n01001013
# text = Para los que sigan las transiciones de las redes sociales de Capitol Hill, esto será algo diferente.
# text_en = For those who follow social media transitions on Capitol Hill, this will be a little different.
1	Para	para	ADP	IN	_	4	case	_	_
2	los	el	DET	DT	Definite=Def|Gender=Masc|Number=Plur|PronType=Art	3	det	_	_
3	que	que	PRON	REL	Gender=Masc|Number=Plur|PronType=Rel	4	nsubj	_	_
4	sigan	_	VERB	VBC	Mood=Sub|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin	17	xcomp	_	_
5	las	el	DET	DT	Definite=Def|Gender=Fem|Number=Plur|PronType=Art	6	det	_	_
6	transiciones	_	NOUN	NN	Gender=Fem|Number=Plur	4	obj	_	_
7	de	de	ADP	IN	_	9	case	_	_
8	las	el	DET	DT	Definite=Def|Gender=Fem|Number=Plur|PronType=Art	9	det	_	_
9	redes	_	NOUN	NN	Gender=Fem|Number=Plur	6	nmod	_	_
10	sociales	_	ADJ	JJ	Gender=Fem|Number=Plur	9	amod	_	_
11	de	de	ADP	IN	_	12	case	_	_
12	Capitol	Capitol	PROPN	NNP	Number=Sing	9	nmod	_	_
13	Hill	hill	NOUN	NN	Number=Sing	12	flat:name	_	SpaceAfter=No|Proper=True
14	,	,	PUNCT	,	_	4	punct	_	_
15	esto	esto	DET	DT	Gender=Masc|Number=Sing|PronType=Dem	17	nsubj	_	_
16	será	ser	AUX	VBC	Mood=Ind|Number=Sing|Person=3|Tense=Fut|VerbForm=Fin	17	cop	_	_
17	algo	algo	NOUN	NN	Gender=Masc|Number=Sing	0	root	_	_
18	diferente	diferente	ADJ	JJ	Gender=Masc|Number=Sing	17	amod	_	SpaceAfter=No
19	.	.	PUNCT	.	_	17	punct	_	_
```

Finally, **meaning**

Computational semantics: key areas

- Formal semantics: Logic, model-theoretic semantics
  - NLTK Book ch.10 Analyzing the meaning of sentences
- Word sense: lexical semantics
  - J&M Ch.23: Word senses and WordNet
  - NLTK Book 2.5 WordNet

Finally, **meaning**

Computational semantics: key areas

- Formal semantics: Logic, model-theoretic semantics
  - NLTK Book ch.10 Analyzing the meaning of sentences
- Word sense: lexical semantics
  - J&M Ch.23: Word senses and WordNet
  - NLTK Book 2.5 WordNet
- Word sense: vector semantics
  - J&M Ch.6: Vector semantics and embeddings
- Predicate-argument semantics, semantic roles
  - J&M Ch.24: Semantic role labeling
  - NLTK how to, PropBank

Vast landscape, so little time…

Word sense: lexical semantics

- A **word sense** is a discrete representation of one aspect of the meaning of a word.
- Challenge: 1 word =/= 1 sense.
  - **Polysemy**: a single word can have multiple meanings (**ambiguity**)
  - **Synonymy**: many words can share the same meaning.
- Computational lexical semantics
  - Goal: Develop, maintain and utilize computerized thesaurus —a database that represents word senses.
  - Focuses on relations between senses
- Meaning relations
  - synonymy
  - antonymy
  - taxonomic relations: hyponym, hypernym
  - meronymy (part-whole)

WordNet

- Project home: https://wordnet.princeton.edu/
- A hierarchical semantic database ("ontology") for English and many other languages.
- Beyond definitions, encodes *relations* between senses.

WordNet

- Project home: https://wordnet.princeton.edu/
- A hierarchical semantic database ("ontology") for English and many other languages.
- Beyond definitions, encodes *relations* between senses.
- Available via NLTK as `nltk.corpus.wordnet`
  - NLTK book https://www.nltk.org/book/ch02.html#wordnet
- **A single unique meaning** is designated as something like 'car.n.01': first noun meaning of 'car'. This is referred to as a **synset**: a "synonym set"
  - The idea: a *unique meaning* is represented by *a set of synonyms* that share the meaning.
  - 'car.n.01' is the most generic meaning of 'car', which can be seen through .definition()

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
    [Synset('car.n.01')]
>>> wn.synset('car.n.01').lemma_names()
    ['car', 'auto', 'automobile', 'machine', 'motorcar']
>>> wn.synset('car.n.01').definition()
    'a motor vehicle with four wheels; usually propelled by an internal combustion
    engine'

>>> wn.synsets('car')
    [Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'),
    Synset('car.n.04'), Synset('cable_car.n.01')]

>>> for syn in wn.synsets('car'):
...     print(syn, syn.lemma_names())

    Synset('car.n.01') ['car', 'auto', 'automobile', 'machine', 'motorcar']
    Synset('car.n.02') ['car', 'railcar', 'railway_car', 'railroad_car']
    Synset('car.n.03') ['car', 'gondola']
    Synset('car.n.04') ['car', 'elevator_car']
    Synset('cable_car.n.01') ['cable_car', 'car']
>>>
```

'car.n.01' represents a **synset** (="synonym set"), a single unique sense.

'car' has 5 distinct senses

Each sense can be conveyed by a set of synonymous words

3 minutes

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')
    [Synset('car.n.01')]
>>> wn.synset('car.n.01').lemma_names()
    ['car', 'auto', 'automobile', 'machine', 'motorcar']
>>> wn.synset('car.n.01').definition()
    'a motor vehicle with four wheels; usually propelled by an internal combustion
    engine'

>>> wn.synsets('car')
    [Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'),
    Synset('car.n.04'), Synset('cable_car.n.01')]

>>> for syn in wn.synsets('car'):
...     print(syn, syn.lemma_names())

    Synset('car.n.01') ['car', 'auto', 'automobile', 'machine', 'motorcar']
    Synset('car.n.02') ['car', 'railcar', 'railway_car', 'railroad_car']
    Synset('car.n.03') ['car', 'gondola']
    Synset('car.n.04') ['car', 'elevator_car']
    Synset('cable_car.n.01') ['cable_car', 'car']
>>>
```

Also try:
`.examples()`

Who can find a word with most synsets? (= most senses)

Wrapping up

- Wednesday: Na-Rae's office hours canceled
- Thursday:
  - Continue computational semantics
  - Over Zoom! Link on Canvas.
- Exercise 11 out
  - Last exercise!