CS-AD 220 – Spring 2016

Natural Language Processing

Session 19: 7-Apr-16

Prof. Nizar Habash

Some slides are adapted from talks by Jurafsky and Martin; Nivre; and Mooney.

NYUAD Course CS-AD 220 – Spring 2016
Natural Language Processing
Assignment #3 : POS Tagging and Parsing
Assigned Mar 31, 2016 / Due Apr 17, 2016 (11:59pm)

I. Grading & Submission

This assignment is about the development of a dependency parser and a part-of-speech (POS) tagger for English. The assignment accounts for 15% of the full grade. It consists of five exercises. **There is also a bonus exercise that can count for up to 5% of the full grade**. The additional exercise consists of a parsing competition on an unseen test set. Participation earns 2%. The first, second and third ranked systems earn additional 3%, 2% and 1%, respectively.

Assignment #3 posted on NYU Classes

START EARLY!

DEADLINE PUSHED FORWARD TO APR 17

Looking ahead

- Prof. Hend Alkhalifa
  - April 14
- Hackathon!
  - April 15-17
- Deadline Assignment #3
  - April 17
- Prof. Jan Hajic
  - April 21

Hackathon Plans

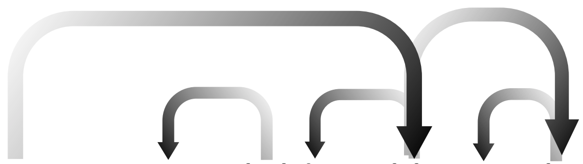
- Diplobot group
- Qusasat group
- Other groups?

Types of Syntactic Analyses

- Phrase Structure Parsing
  - aka constituency parsing
  - CKY Parser
- **Dependency Parsing**
  - **MaltParser**
- Chunking
  - Base-phrase chunking

MaltParser Example

Actions	Stack	Buffer	Arcs A	*Pre-conditions*
Shift	σ	$w_i \mid \beta$	A	
➔	$\sigma \mid w_i$	β	A	
Reduce	$\sigma \mid w_i$	β	A	$r'(w_k, w_i) \in A$
➔	σ	β	A	
Left-Arc$_r$	$\sigma \mid w_i$	$w_j \mid \beta$	A	$r'(w_k, w_i) \notin A$, $w_i \neq ROOT$
➔	σ	$w_j \mid \beta$	$A \cup \{r(w_j, w_i)\}$	e.g., children |-| want
Right-Arc$_r$	$\sigma \mid w_i$	$w_j \mid \beta$	A	
➔	$\sigma \mid w_i \mid w_j$	β	$A \cup \{r(w_i, w_j)\}$	e.g., want |-| toys

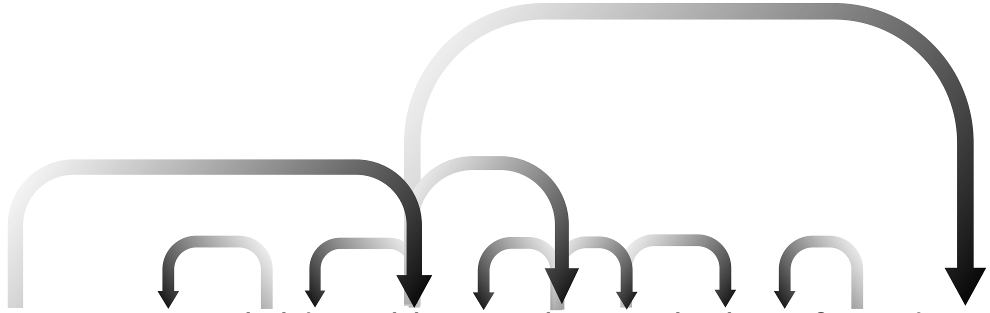


ROOT Happy children like to play with their friends .

	[ROOT]	[Happy, children, …]	$\varnothing$
Shift	[ROOT, Happy]	[children, like, …]	$\varnothing$
LA$_{amod}$	[ROOT]	[children, like, …]	{amod(children, happy)} = A_1
Shift	[ROOT, children]	[like, to, …]	A_1
LA$_{nsubj}$	[ROOT]	[like, to, …]	$A_1 \cup$ {nsubj(like, children)} = A_2
RA$_{root}$	[ROOT, like]	[to, play, …]	$A_2 \cup$ {root(ROOT, like) = A_3
Shift	[ROOT, like, to]	[play, with, …]	A_3
LA$_{aux}$	[ROOT, like]	[play, with, …]	$A_3 \cup$ {aux(play, to) = A_4
RA$_{xcomp}$	[ROOT, like, play]	[with their, …]	$A_4 \cup$ {xcomp(like, play) = A_5

MaltParser Example

Actions	Stack	Buffer	Arcs A	*Pre-conditions*
Shift	σ	$w_i \mid \beta$	A	
➔	$\sigma \mid w_i$	β	A	
Reduce	$\sigma \mid w_i$	β	A	$r'(w_k, w_i) \in A$
➔	σ	β	A	
Left-Arc$_r$	$\sigma \mid w_i$	$w_j \mid \beta$	A	$r'(w_k, w_i) \notin A$, $w_i \neq$ ROOT
➔	σ	$w_j \mid \beta$	$A \cup \{r(w_j, w_i)\}$	e.g., children|-| want
Right-Arc$_r$	$\sigma \mid w_i$	$w_j \mid \beta$	A	
➔	$\sigma \mid w_i \mid w_j$	β	$A \cup \{r(w_i, w_j)\}$	e.g., want |-| toys

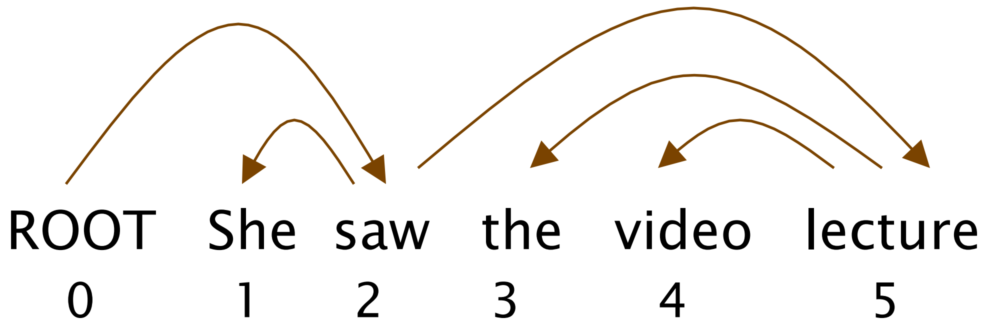


ROOT Happy children like to play with their friends .

RA$_{xcomp}$	[ROOT, like, play]	[with their, …]	$A_4 \cup$ {xcomp(like, play) = A_5
RA$_{prep}$	[ROOT, like, play, with]	[their, friends, …]	$A_5 \cup$ {prep(play, with) = A_6
Shift	[ROOT, like, play, with, their]	[friends, .]	A_6
LA$_{poss}$	[ROOT, like, play, with]	[friends, .]	$A_6 \cup$ {poss(friends, their) = A_7
RA$_{pobj}$	[ROOT, like, play, with, friends]	[.]	$A_7 \cup$ {pobj(with, friends) = A_8
Reduce	[ROOT, like, play, with]	[.]	A_8
Reduce	[ROOT, like, play]	[.]	A_8
Reduce	[ROOT, like]	[.]	A_8
RA$_{punc}$	[ROOT, like, .]	[]	$A_8 \cup$ {punc(like, .) = A_9

Terminate as soon as the buffer is empty. Dependencies = A_9

Evaluation: Dependency Accuracy



$$\text{Acc} = \frac{\#\text{ correct deps}}{\#\text{ of deps}}$$

UAS =

LAS =

Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	dobj

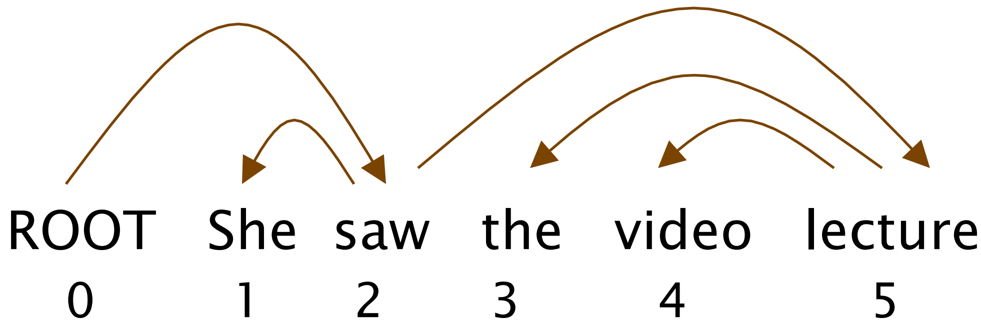
Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

UAS = Unlabeled Attachment Score

LAS = Labeled Attachment Score

Evaluation: Dependency Accuracy



$$\text{Acc} = \frac{\#\ \text{correct deps}}{\#\ \text{of deps}}$$

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	dobj

Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

UAS = Unlabeled Attachment Score

LAS = Labeled Attachment Score

Representative performance numbers

- The CoNLL-X (2006) shared task provides evaluation numbers for various dependency parsing approaches over 13 languages
  - MALT: LAS scores from 65–92%, depending greatly on language/treebank
- Here we give a few UAS numbers for English to allow some comparison to constituency parsing

Parser	UAS%
Sagae and Lavie (2006) ensemble of dependency parsers	92.7
Charniak (2000) generative, constituency	92.2
Collins (1999) generative, constituency	91.7
McDonald and Pereira (2005) – MST graph-based dependency	91.5
Yamada and Matsumoto (2003) – transition-based dependency	90.4

Projectivity

- Dependencies from a CFG tree using heads, must be projective
  - There must not be any crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words.
- But dependency theory normally does allow non-projective structures to account for displaced constituents
  - You can't easily get the semantics of certain constructions right without these nonprojective dependencies

Handling non-projectivity

- The arc-eager algorithm we presented only builds projective dependency trees
- Possible directions to head:
  1. Just declare defeat on non-projective arcs
  2. Use a dependency formalism which only admits projective representations (a CFG doesn't represent such structures…)
  3. Use a postprocessor to a projective dependency parsing algorithm to identify and resolve non-projective links
  4. Add extra types of transitions that can model at least most non-projective structures
  5. Move to a parsing mechanism that does not use or require any constraints on projectivity (e.g., the graph-based MSTParser)

Types of Syntactic Analyses

- Phrase Structure Parsing
  - aka constituency parsing
  - CKY Parser
- Dependency Parsing
  - MaltParser
- **Chunking**
  - **Base-phrase chunking**

Phrase Chunking

- Find all non-recursive noun phrases (NPs) and verb phrases (VPs) in a sentence.
  - Base phrases
  - [NP I] [VP ate] [NP the spaghetti] [PP with] [NP meatballs].
  - [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September]

Phrase Chunking as Sequence Labeling
IOB Chunkers

- Tag individual words with one of 3 tags
  - B (Begin) word starts new target phrase
  - I (Inside) word is part of target phrase but not the first word
  - O (Other) word is not part of target phrase
- Sample for NP chunking
  - He reckons the current account deficit will narrow to only # 1.8 billion in September.

Begin **Inside** **Other**

Example

(from YAMCHA: Yet Another Multipurpose Chunk Annotator)

He	PRP	B-NP
reckons	VBZ	B-VP
the	DT	B-NP
current	JJ	I-NP
account	NN	I-NP
deficit	NN	I-NP
will	MD	B-VP
narrow	VB	I-VP
to	TO	B-PP
only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
billion	CD	I-NP
in	IN	B-PP
September	NNP	B-NP
.	.	O

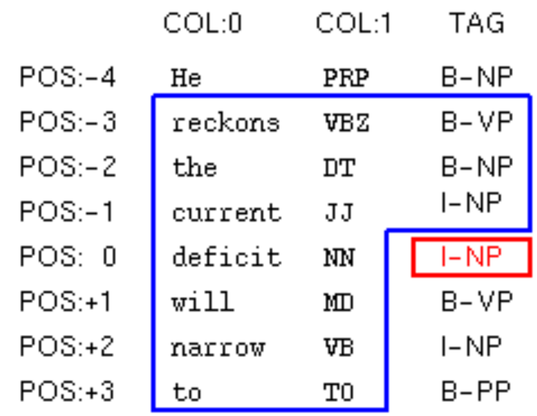
http://chasen.org/~taku/software/yamcha/

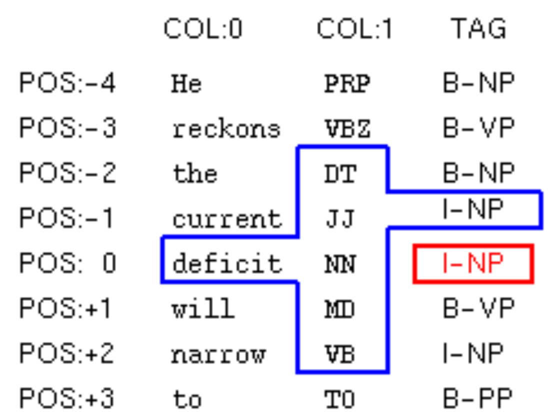
Example

(from YAMCHA: Yet Another Multipurpose Chunk Annotator)

He	PRP	B-NP
reckons	VBZ	B-VP
the	DT	B-NP
current	JJ	I-NP
account	NN	I-NP
deficit	NN	I-NP
will	MD	B-VP
narrow	VB	I-VP
to	TO	B-PP
only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
billion	CD	I-NP
in	IN	B-PP
September	NNP	B-NP
.	.	O

Some of the possible feature sets that can be used to classify tags





http://chasen.org/~taku/software/yamcha/

Evaluating Chunking

- Per token accuracy does not evaluate finding correct full chunks. Instead use:

$$\text{Precision} = \frac{\text{Number of correct chunks found}}{\text{Total number of chunks found}}$$

$$\text{Recall} = \frac{\text{Number of correct chunks found}}{\text{Total number of actual chunks}}$$

- Take harmonic mean to produce a single evaluation metric called F measure.

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \qquad F_1 = \frac{1}{(\frac{1}{P} + \frac{1}{R})/2} = \frac{2PR}{P + R}$$

Current Chunking Results

- English NP chunking performance: $F_1 \sim 96\%$
- Typical results for finding range of chunk types (CONLL 2000 shared task: NP, VP, PP, ADV, SBAR, ADJP) is F_1=92−94%

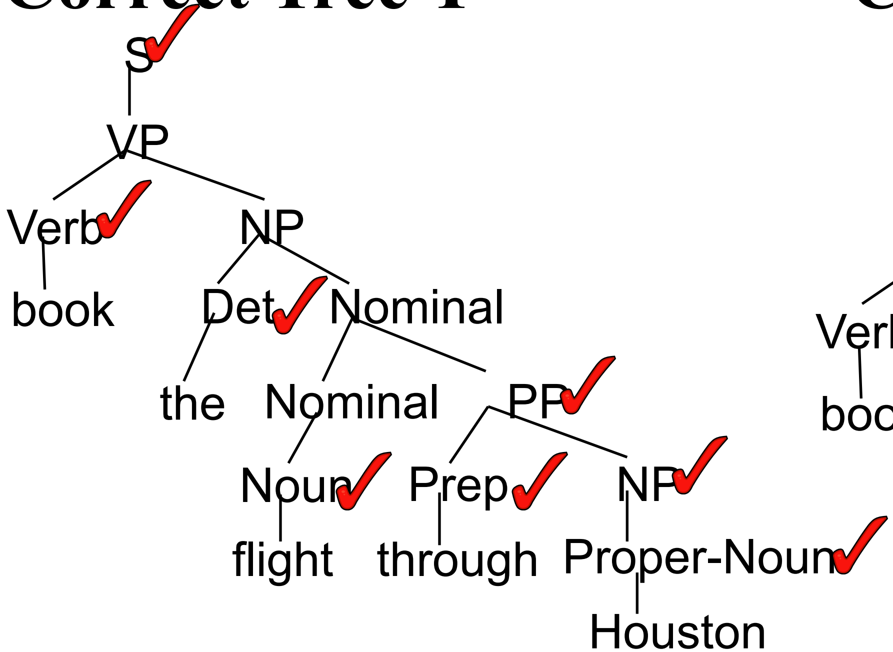
Constituency Parsing Evaluation

- PARSEVAL metrics measure the fraction of the constituents that match between the computed and human parse trees. If *P* is the system's parse tree and *T* is the human parse tree (the "gold standard"):
  - ***Recall*** = (# correct constituents in *P*) / (# constituents in *T*)
  - ***Precision*** = (# correct constituents in *P*) / (# constituents in *P*)
- ***Labeled precision*** and ***labeled recall*** require getting the non-terminal label on the constituent node correct to count as correct.
- F_1 is the harmonic mean of precision and recall.

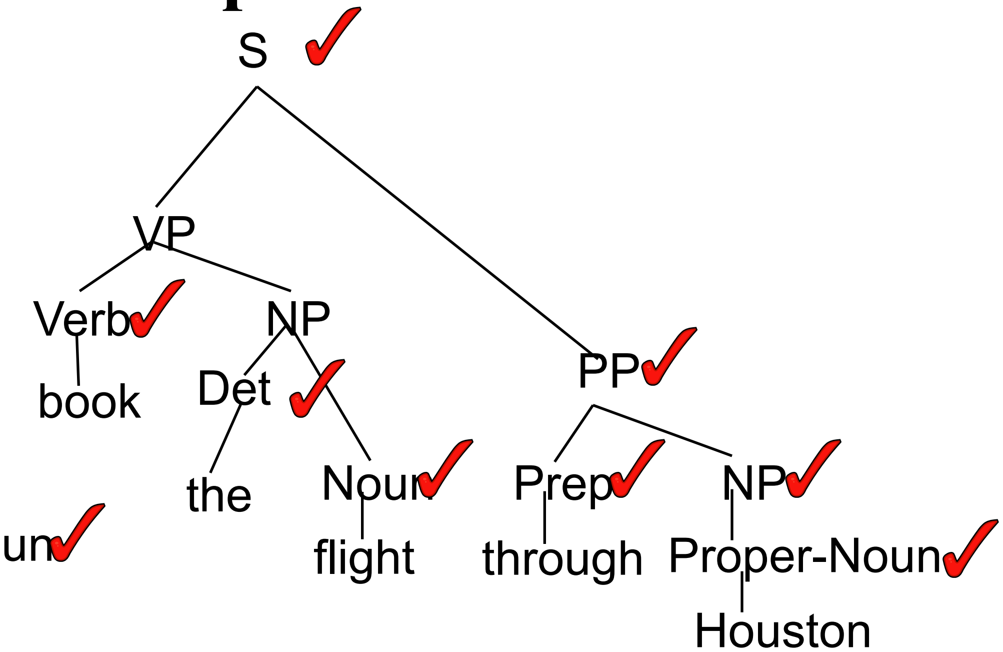
English labeled precision and recall number are slightly above 90%

Computing Evaluation Metrics

Correct Tree T



Computed Tree P



Constituents: 12

Constituents: 10

Correct Constituents: 8

Recall = 8/12= 66.7%   Precision = 8/10=80.0% F_1 = 72.7%

$$
\begin{tikzcd}
& U \arrow[dl] \arrow[d] \arrow[dr] & \\
Q & I & Z
\end{tikzcd}
$$

Quiz #4

Name:________________

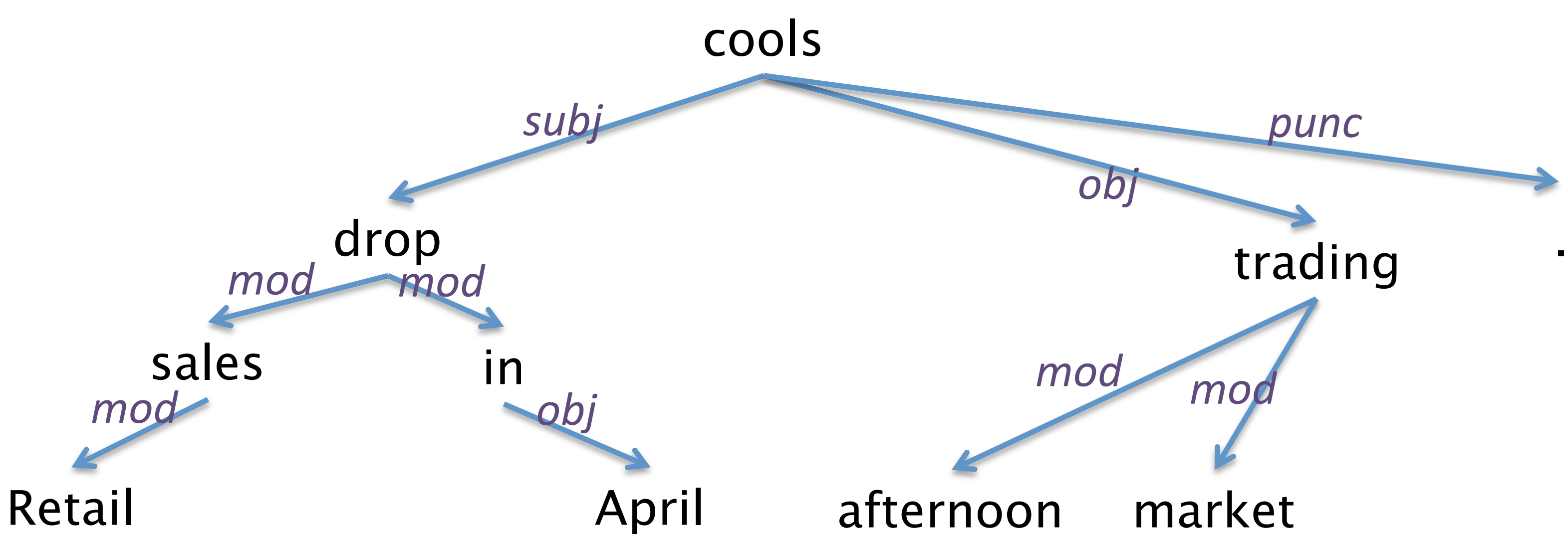
Create a dependency parse tree for this sentence. Use the following four relationship labels: subj, obj, mod, punc.

Retail sales drop in April cools afternoon market trading.

Quiz #4

Name:________________

Create a dependency parse tree for this sentence. Use the following four relationship labels: subj, obj, mod, punc.



Retail sales drop in April cools afternoon market trading.

Next Time

- Read J+M Chap 25 (intro up to 25.5)
- Come with questions about Assignment #3