CS 4650/7650:
Natural Language Processing

Midterm Review & Dependency Parsing Grammar

Diyi Yang

Many slides from Yulia Tsvetkov, Greg Durrett, David Bamman, and others

Quick Info on Midterm

Quick Info on Project

- Team Information: March 04th
- Project Proposal: March 11th
- Midway Report: Apr 1st
- Poster: Apr 17th
- Poster Presentation: Apr 20th
- Final Report: Apr 23rd

Please Submit Your Team Info

Team Information: March 04th

bit.ly/3ahYtxJ

Quick Info on Project

- Team Information: March 04th
- Project Feedback: March 9th (In-Class)
- Project Proposal: March 11th
- Midway Report: Apr 1st
- Poster: Apr 17th
- Poster Presentation: Apr 20th
- Final Report: Apr 23rd

Project Ideas

- https://docs.google.com/document/d/1cWZDwdFG-XwPAVIByN3FBRMa8n7MZulfL4dcOw07AV8/edit

Recap: Context-free Grammar

- A context-free grammar defines how symbols in a language combine to form valid structures

NP	→	Det Nominal
NP	→	ProperNoun
Nominal	→	Noun | Nominal Noun
Det	→	a | the
Noun	→	flight

non-terminals

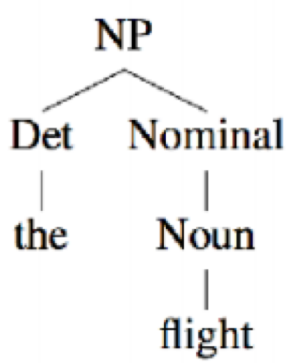
lexicon/ terminals

Recap: Context-free Grammar

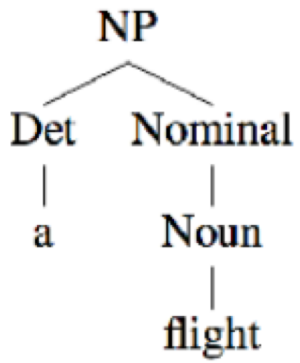
N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ $\beta \in (\Sigma, N)$	$S \rightarrow$ NP VP Noun $\rightarrow$ dog
S	Start symbol	

Recap: Derivation

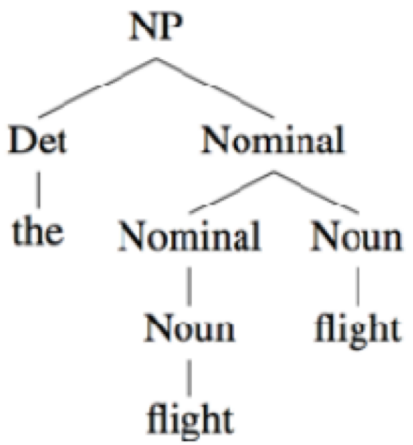
Given a CFG, a derivation is the sequence of productions used to generate a string of words (e.g., a sentence), often visualized as a parse tree



the flight



a flight



the flight flight

Recap: CKY Algorithm

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
      for all {A | A → words[j] ∈ grammar}
              table[j − 1, j] ← table[j − 1, j] ∪ A
      for i ← from j − 2 downto 0 do
          for k ← i + 1 to j − 1 do
              for all {A | A → BC ∈ grammar and B ∈ table[i, k] and C ∈ table[k, j]}
                      table[i,j] ← table[i,j] ∪ A
```

Figure 13.5 The CKY algorithm.

Recap: PCFG

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, a
R	Set of production rules, each $A \rightarrow \beta$ [p] $p = P(\beta \mid A)$	$S \rightarrow NP\ VP$ $Noun \rightarrow dog$
S	Start symbol	

Recap: CKY PCFG

- A PCFG gives us a mechanism for assigning scores (here, probabilities) to different parses for the same sentence
- But we often care about is finding **the single best parse** with the highest probability
- In general, when people talk about CKY, they mean the PCFG version, since the CFG version isn't very useful

Recap: CKY PCFG

- We calculate the max probability parse using CKY by storing the probability of each phrase within each cell as we built it up.
- In particular, we fill the cell for span i-j and label A with the maximum over splits and rules of

$$table(i, j, A) = P(A \rightarrow BC) \times table(i, k, B) \times table(k, j, C)$$

function PROBABILISTIC-CKY(*words*,*grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar \}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$ **and** $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k, B, C\}$
  return BUILD_TREE($back$[1, LENGTH(*words*), S]), $table$[1, LENGTH(*words*), S]

Recap: Ways to Learn PCFG Rule Probabilities

- Penn Treebank

$$P(\alpha \rightarrow \beta|\alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma}\text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- Chicken-and-egg problem

More about PCFG

- Poor impendence assumptions missing rule dependencies
- Lack of sensitivity to lexical dependencies

Parsing Evaluation

- **Intrinsic Evaluation**
  - Automatic: evaluate against annotation provided by human experts (gold standard) according to some predefined measure
  - Manual: … according to human judgment
- **Extrinsic Evaluation**: score syntactic representation by comparing how well a system using this representation performs on some tasks
  - E.g., use syntactic representation as input for a semantic analyzer and compare results of the analyzer using syntax predicted by different parsers

Standard Evaluation Setting in Parsing

- Automatic intrinsic evaluation is used: parsers are evaluated against gold standard by provided by linguists

- There is a standard split into the parts:
  - Training set: used for estimation of model parameters
  - Development set: used for tuning the model (initial experiments)
  - Test set: final experiments to compare against previous work

Automatic Evaluation of Constituent Parsers

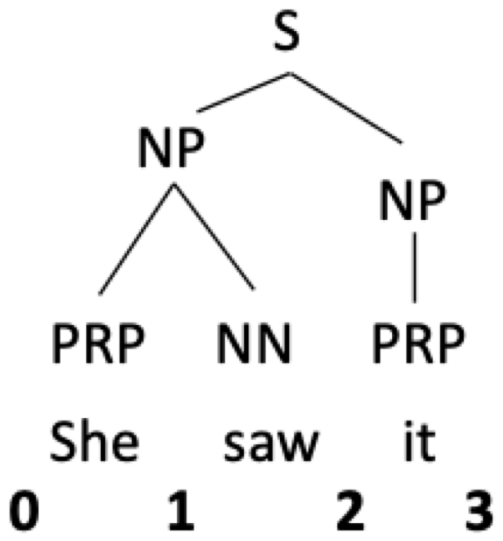
- **Exact match:**
  - percentage of trees predicted correctly
- **Bracket score:**
  - scores how well individual phrases (& their boundaries) are identified

Most standard measure

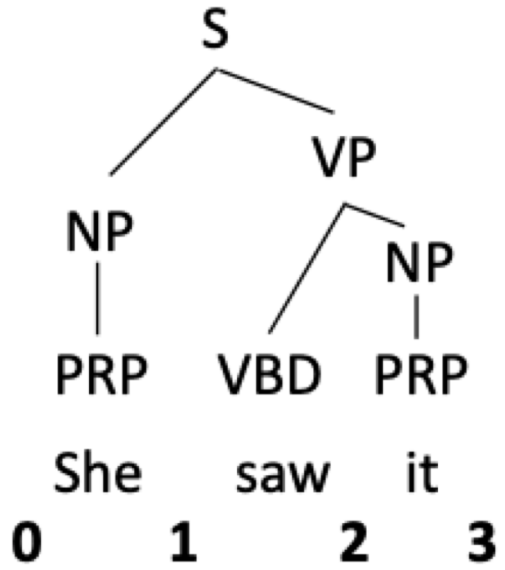
Brackets Scores

- The most standard score is bracket score
- It regards a tree as a collection of brackets: [min, max, C]
- The set of brackets predicted by a parser is compared against the set of brackets in the tree annotated by a linguist
- Precision, recall and F1 are used as scores

Brackets Scores Evaluation



S(0,3),
NP(0,2),
NP(2,3),
~~PRP(0,1),~~
~~NN(1,2),~~
~~PRP(2,3)~~



S(0,3),
NP(0,1),
VP(1,3),
NP(2,3),
~~PRP(0,1),~~
~~VBD(1,2),~~
~~PRP(2,3)~~

- Precision: number of correct brackets / num pred brackets = 2/3
- Recall: number of correct brackets / num of gold brackets = 2/4
- F1: harmonic mean of precision and recall = $(1/2 * ((2/4)^{-1} + (2/3)^{-1}))^{-1}$

$= 0.57$

Example from Greg Durrett

Results

- Standard dataset for English: Penn Treebank (Marcus et al., 1993)
  - Evaluation: F1 over labeled constituents of the sentence
- Vanilla PCFG: ~75 F1
- Best PCFGs for English: ~90 F1
- SOTA (discriminative models): 95 F1
- Other language
  - Results vary widely depending on annotation + complexity of the grammar

Example from Greg Durrett

Efficiency

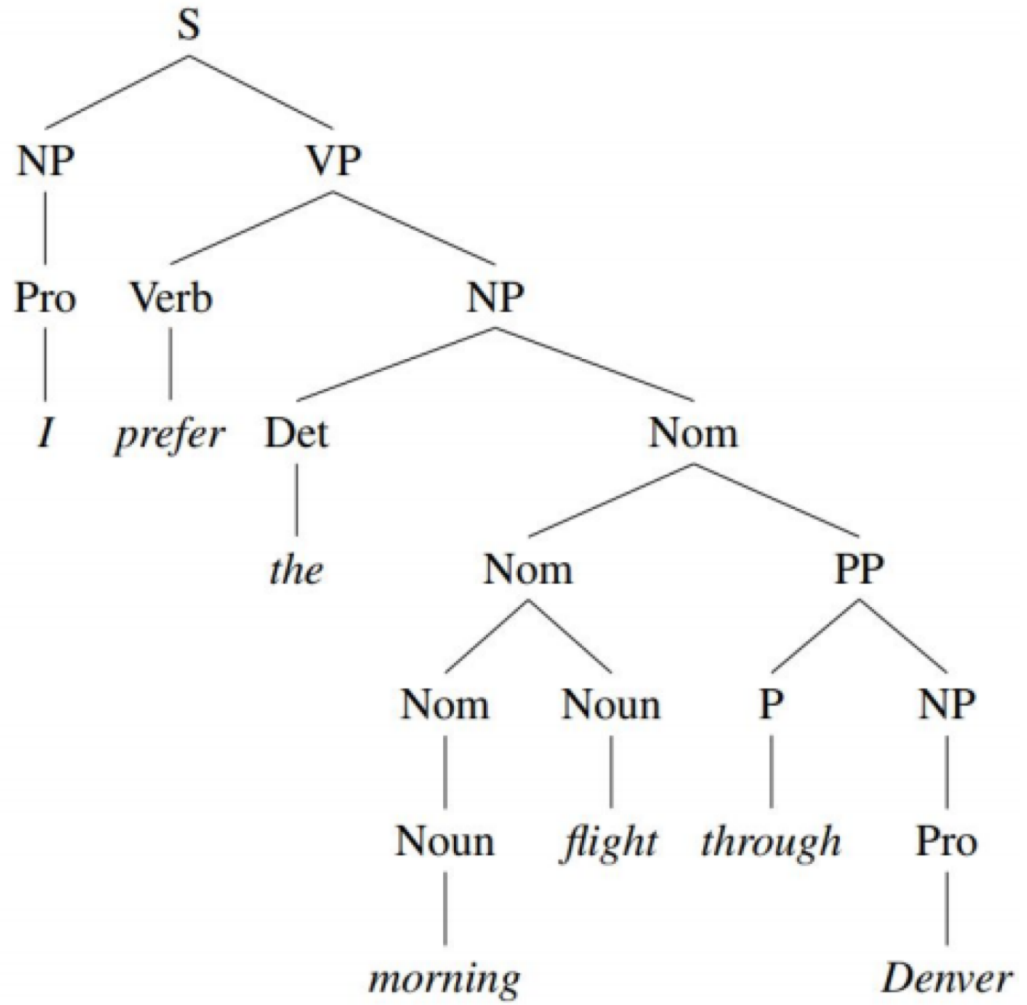
- CKY / PCKY is $|G|*N^3$
  - Grammar can be huge
  - Grammar can be extremely ambiguous
- We care about best parses
- How to improve efficiency?

Efficiency

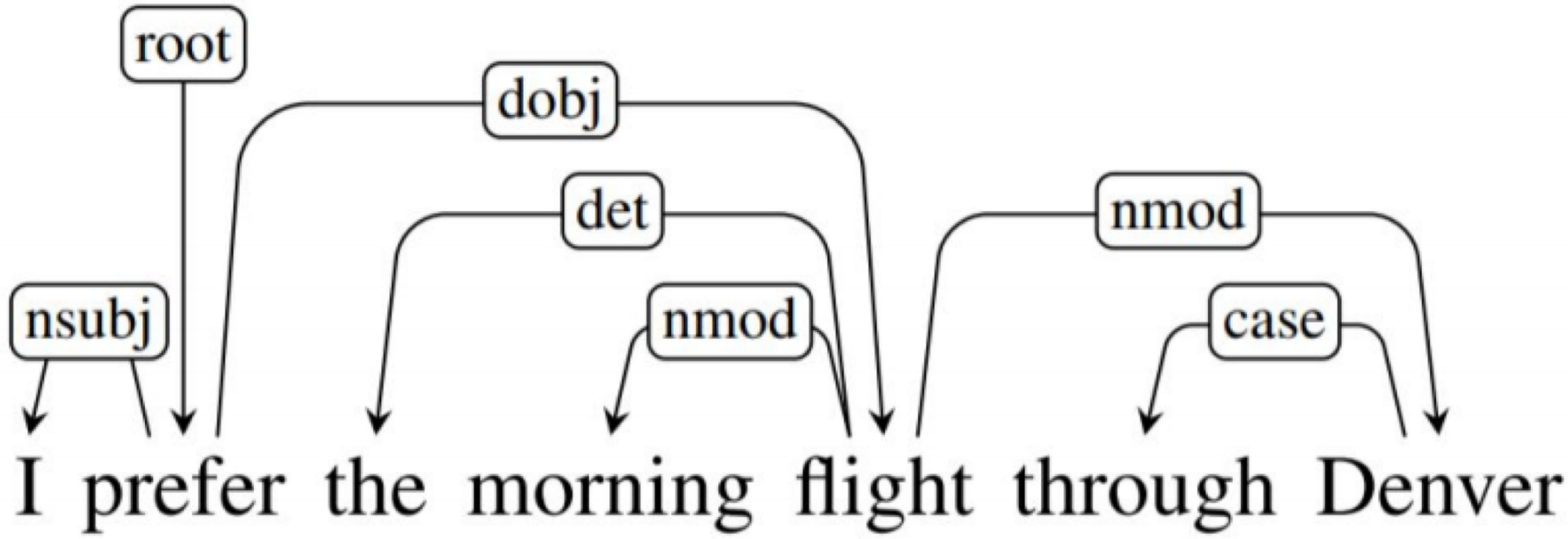
- Beam Search:
  - Assume low probability partial parses unlikely to yield high probability overall
  - Keep only top k most probably partial parses (l=5, 10, 50, etc)
- Heuristic Filtering
  - Some rules/partial parses are unlikely to end up in best parses. Don't store it
    - Low frequency: exclude singleton productions
    - Low probability: exclude constituents x: $p(x) < 10^{-200}$

- Dependency Parsing Grammar

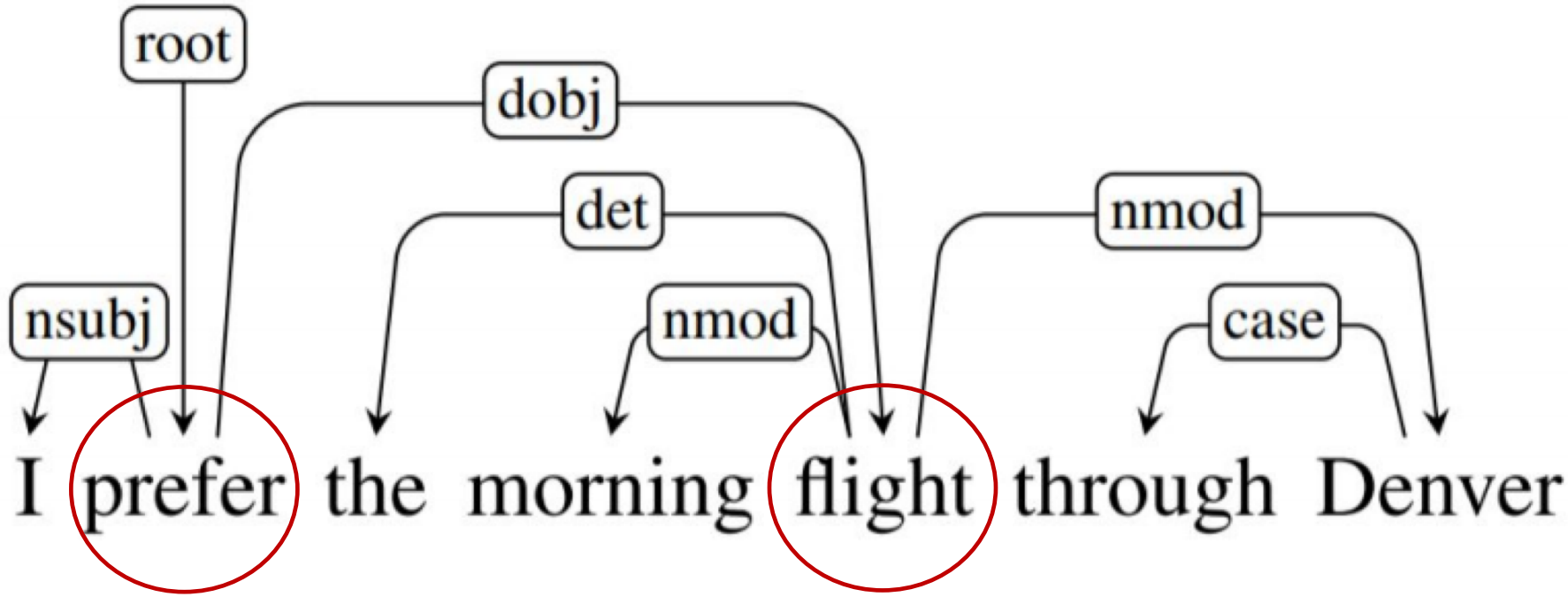
Constituent (Phrase-Structure) Representation



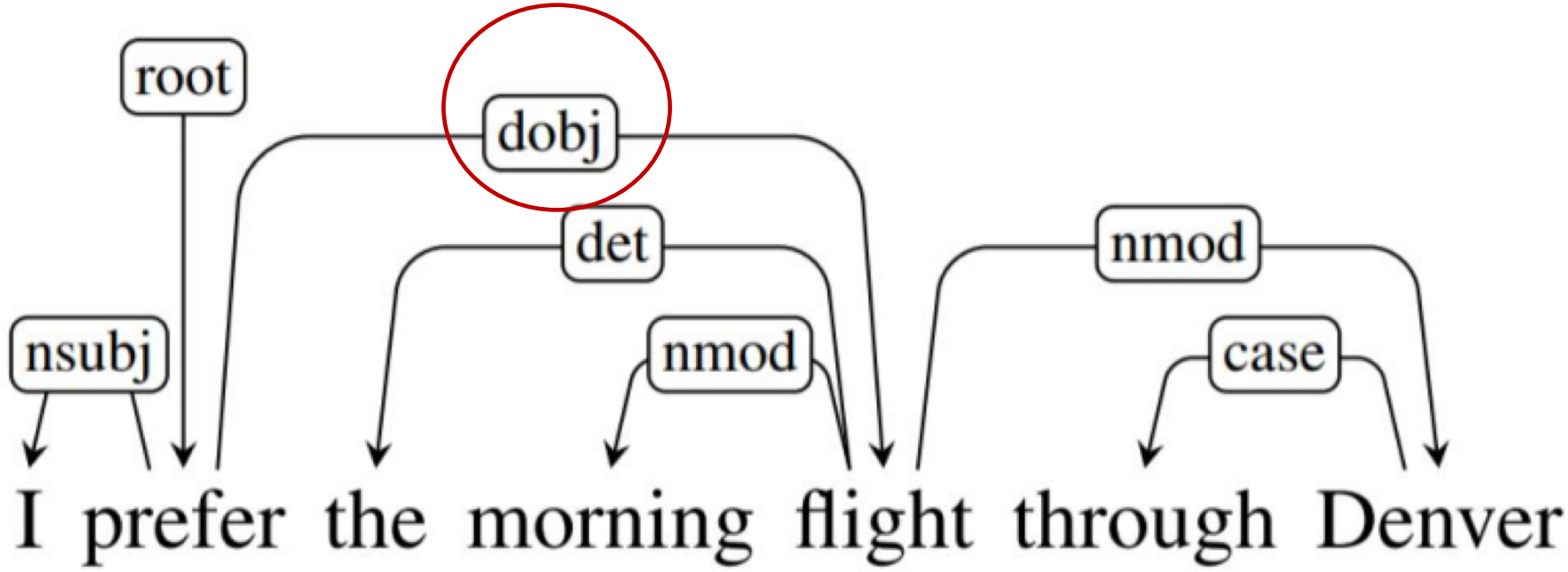
Dependency Representation



Head and Dependent



Grammatical Function



Universal Dependencies (NIVRE ET AL., 2016)

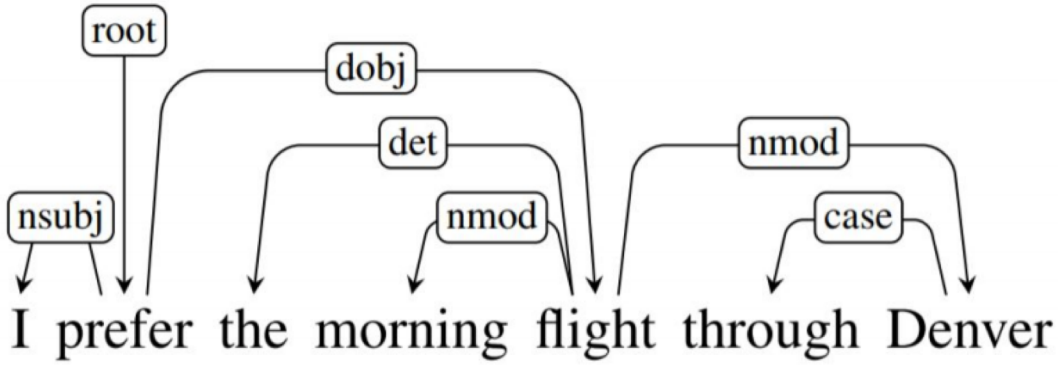
- Developing cross-linguistically consistent treebank annotation for many languages
- Goals:
  - Facilitating multi-lingual parser development
  - Cross-lingual learning
  - Parsing research from a language typology perspective

Grammatical Functions

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Selected dependency relations from the Universal Dependency Set

Two Core Types of Relationships

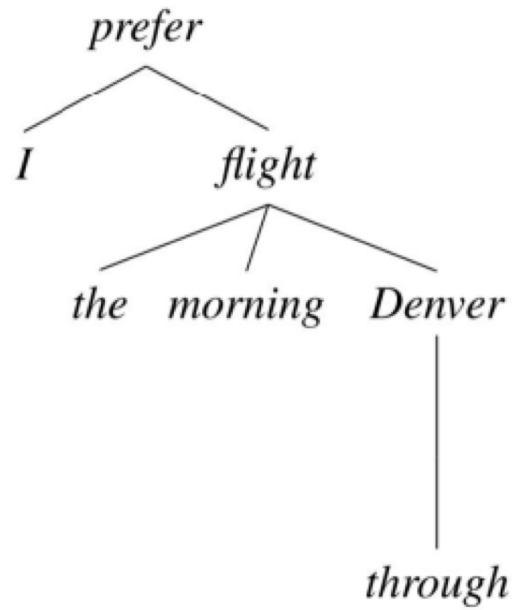


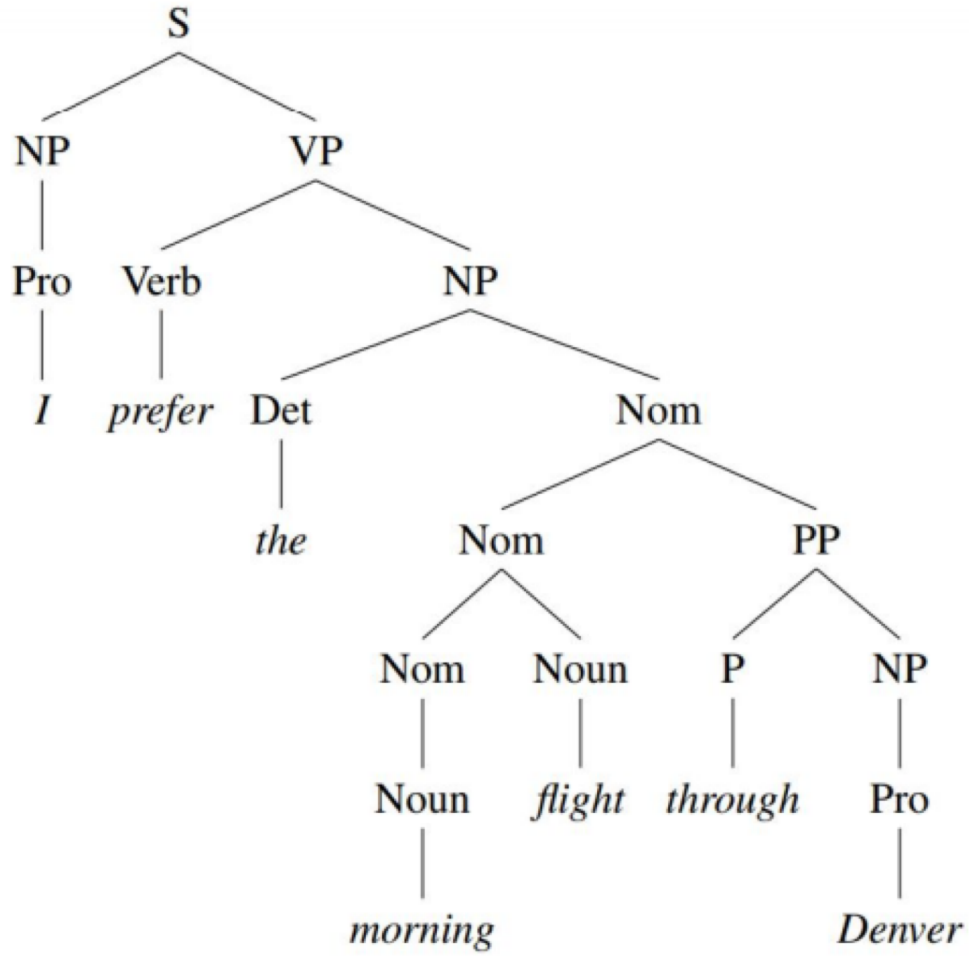
- The clausal relations NSUBJ and DOBJ identify the **arguments**: the subject and direct object of the predicate *cancel*
- The NMOD, DET, and CASE relations denote **modifiers** of the nouns *flights* and *Houston*.

Examples of Core Universal Dependency Relations

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Dependency vs Constituency Trees

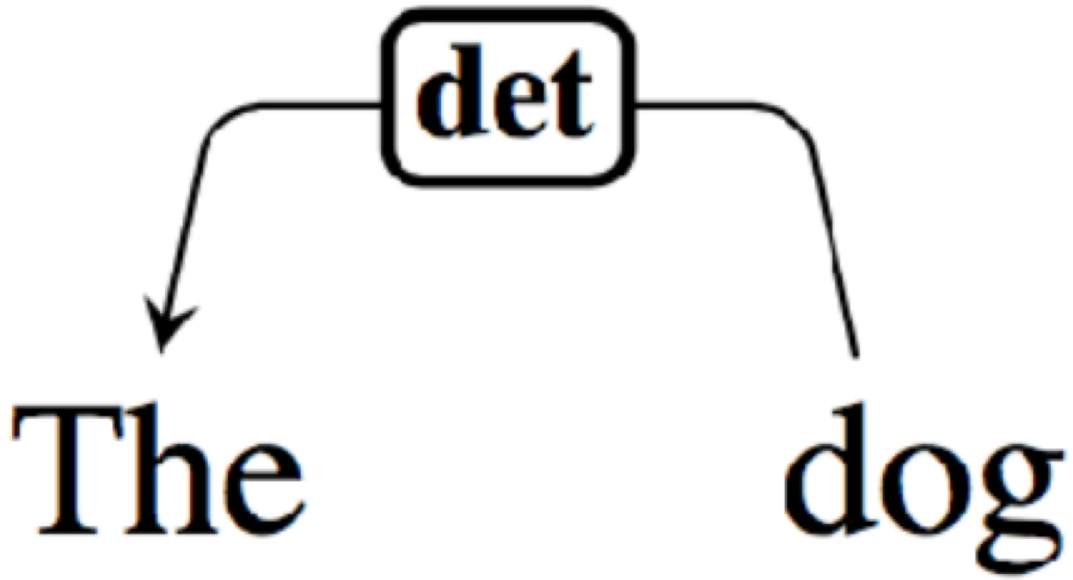


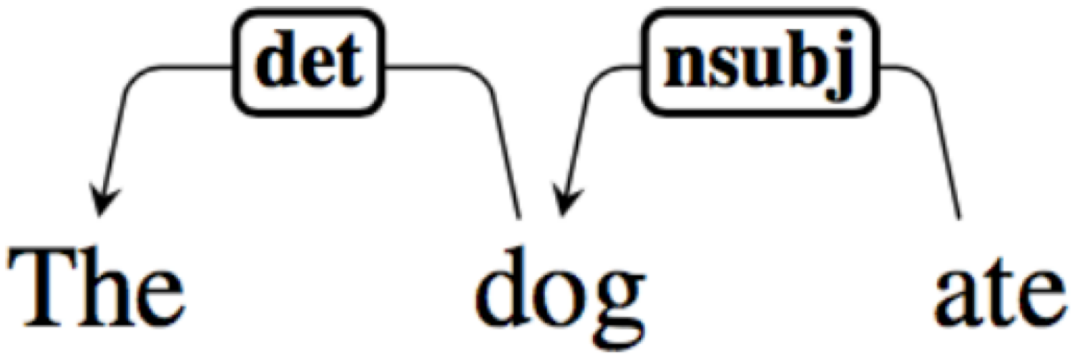


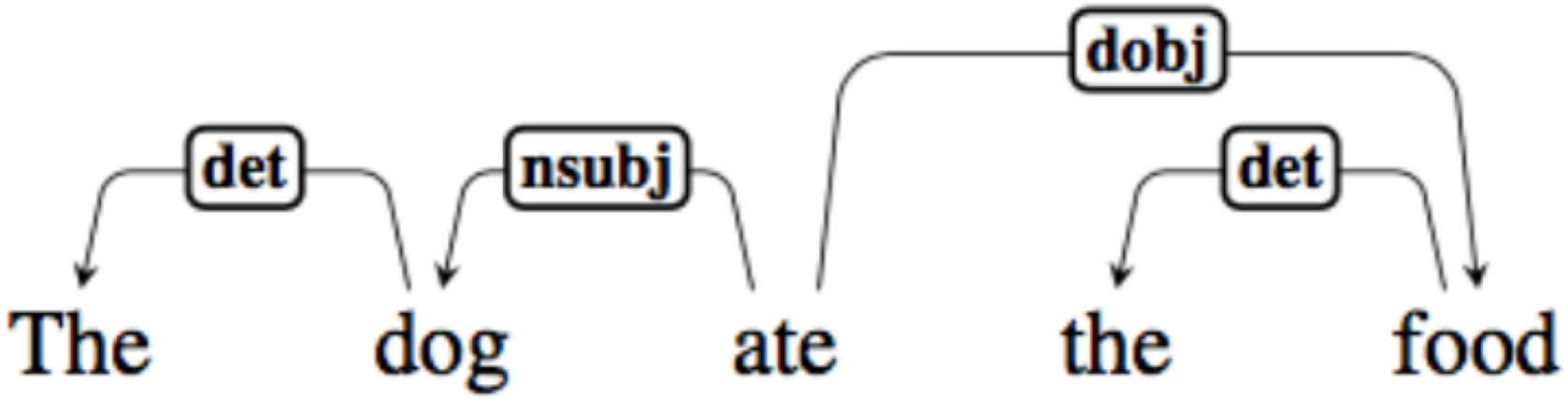
Dependency syntax doesn't have non-terminal structure like a CFG; words are directly linked to each other.

Dependency Syntax

- Syntactic structure = asymmetric, binary relations between words


det
The
dog


det
nsubj
The
dog
ate


dobj
det
nsubj
det
The
dog
ate
the
food

Dependency Representation (Tree/Directed Graphs)

A dependency structure can be defined as a directed graph G, consisting of

- A set V of nodes – words, punctuation, stems, affixes
- A set A of arcs – directed edges

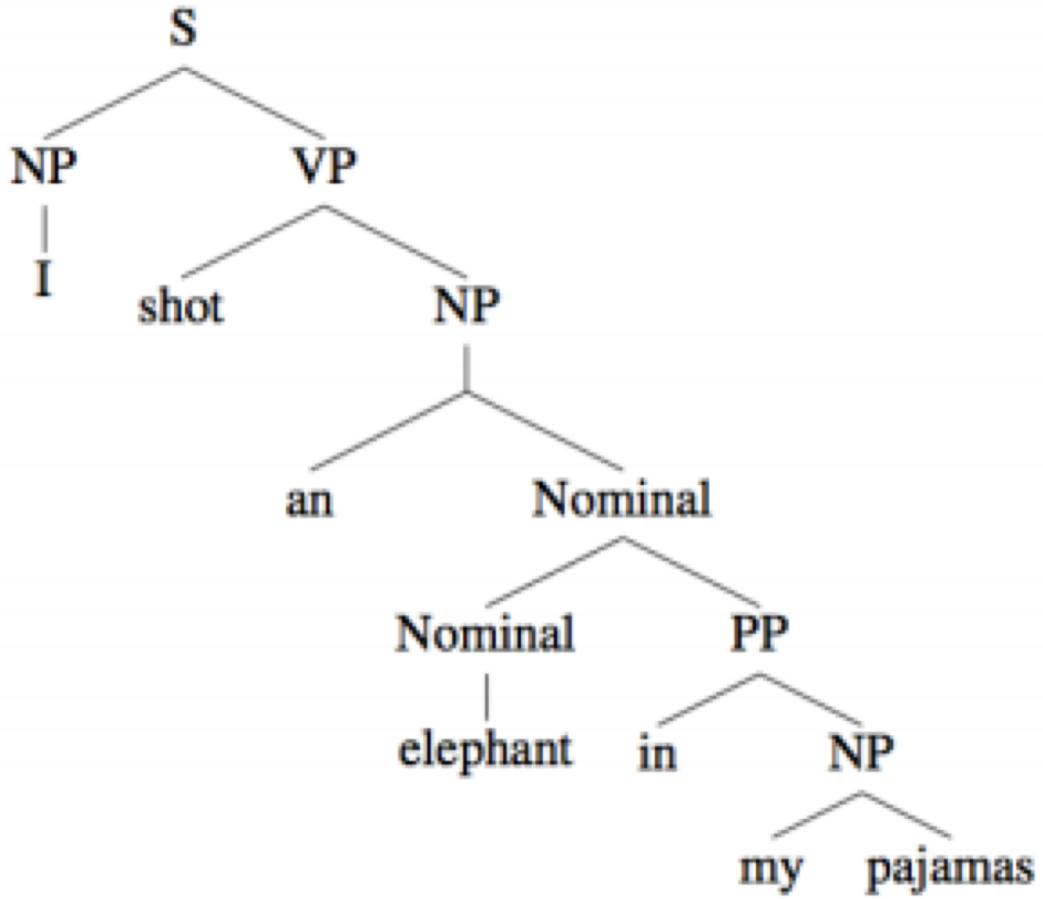
Dependency Representation (Tree)

A dependency structure is typically constrained to form a tree:

- Single root vertex with no incoming arcs
- Every vertex has exactly one incoming arc except root **(single head constraint)**
- There is a unique path from the root to each vertex in V **(acyclic constraint)**

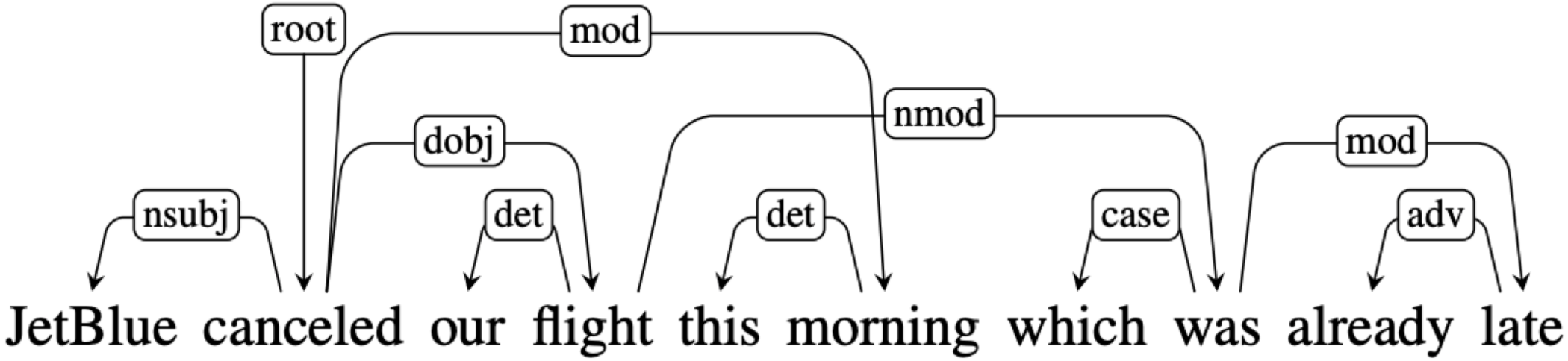
Word Order

- Dependency relations belong to the structural order of a sentence, not the linear order
- This is different from a phrase-structure tree, where the syntax is constrained by the linear order of the sentence (a different linear order yields a different parse tree)



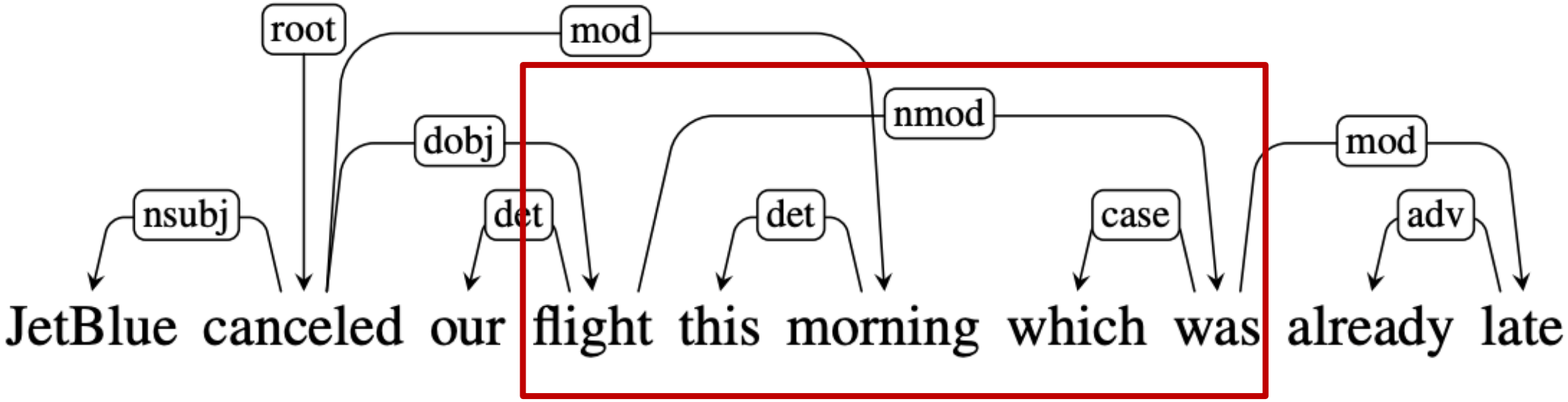
Projectivity

- An arc between a head and dependent is projective if there is a path from the head to every word between the head and dependent.



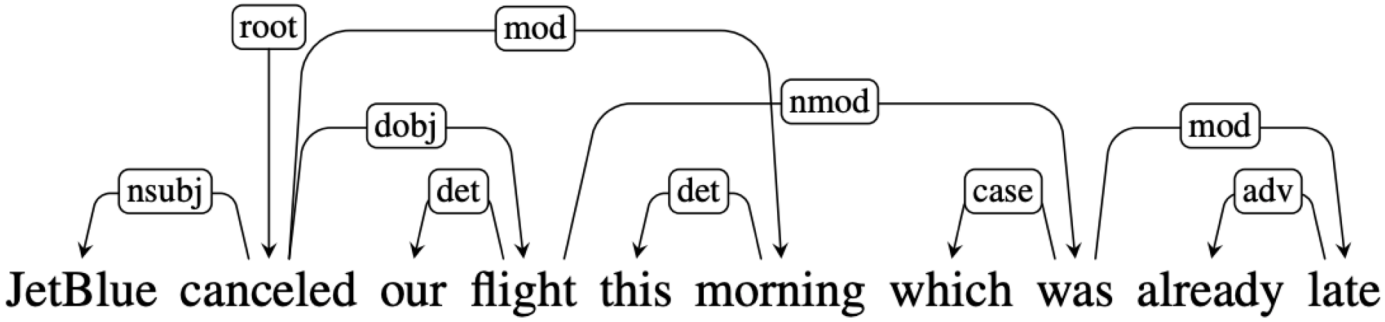
Projectivity

- An arc between a head and dependent is projective if there is a path from the head to every word between the head and dependent.



Projectivity

- Projective parse
  - **Arcs don't across each other**
  - Mostly true for English
- Non-projective structures are needed to account for
  - Long-distance dependencies
  - Flexible word order



Projectivity

- Dependency grammars do not normally assume that all dependency-trees are projective, because some linguistic phenomena can only be achieved using non-projective trees.

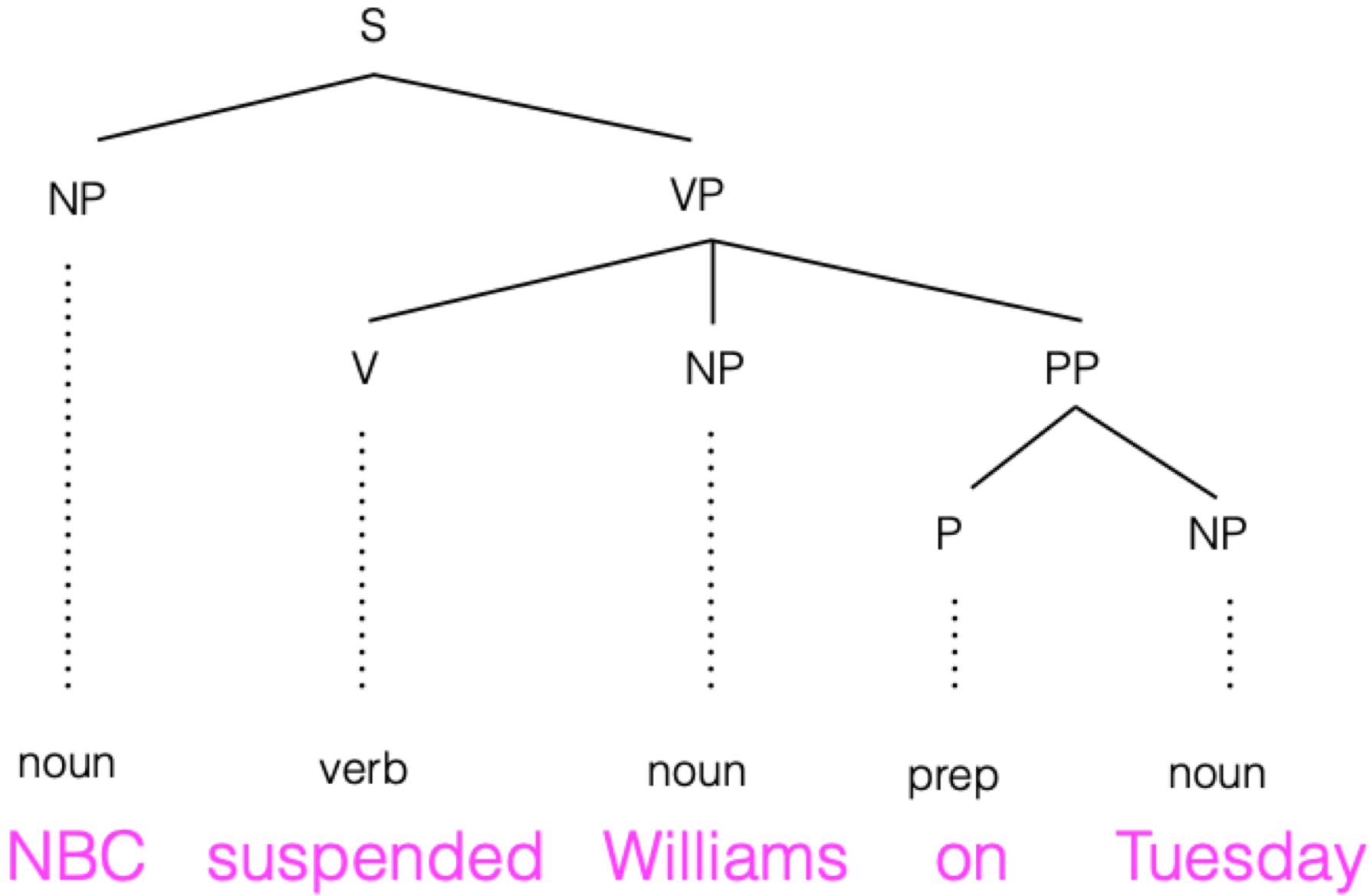
- But a lot of parsers assume that the output trees are projective

- Reasons:
  - Conversion from constituency to dependency
  - The most widely used families of parsing algorithms impose projectivity

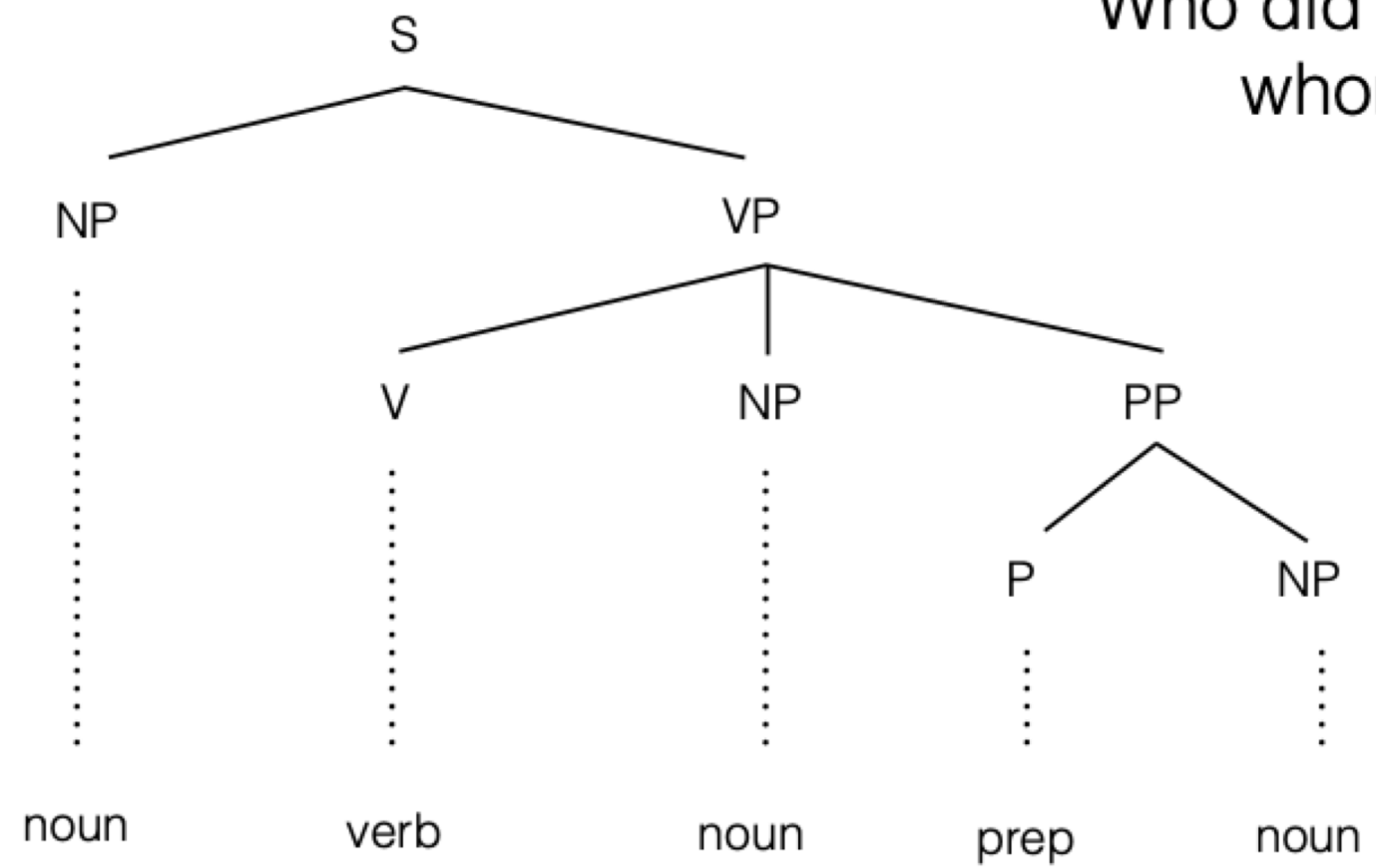
Dependency vs Constituency

- Dependency structures explicitly represent
  - Head-dependent relations (directed arcs)
  - Functional categories (arc labels)
  - Possibly some structural categories (parts of speech)

- Phrase (aka constituent) structures explicitly represent
  - Phrases (nonterminal nodes)
  - Structural categories (nonterminal labels)

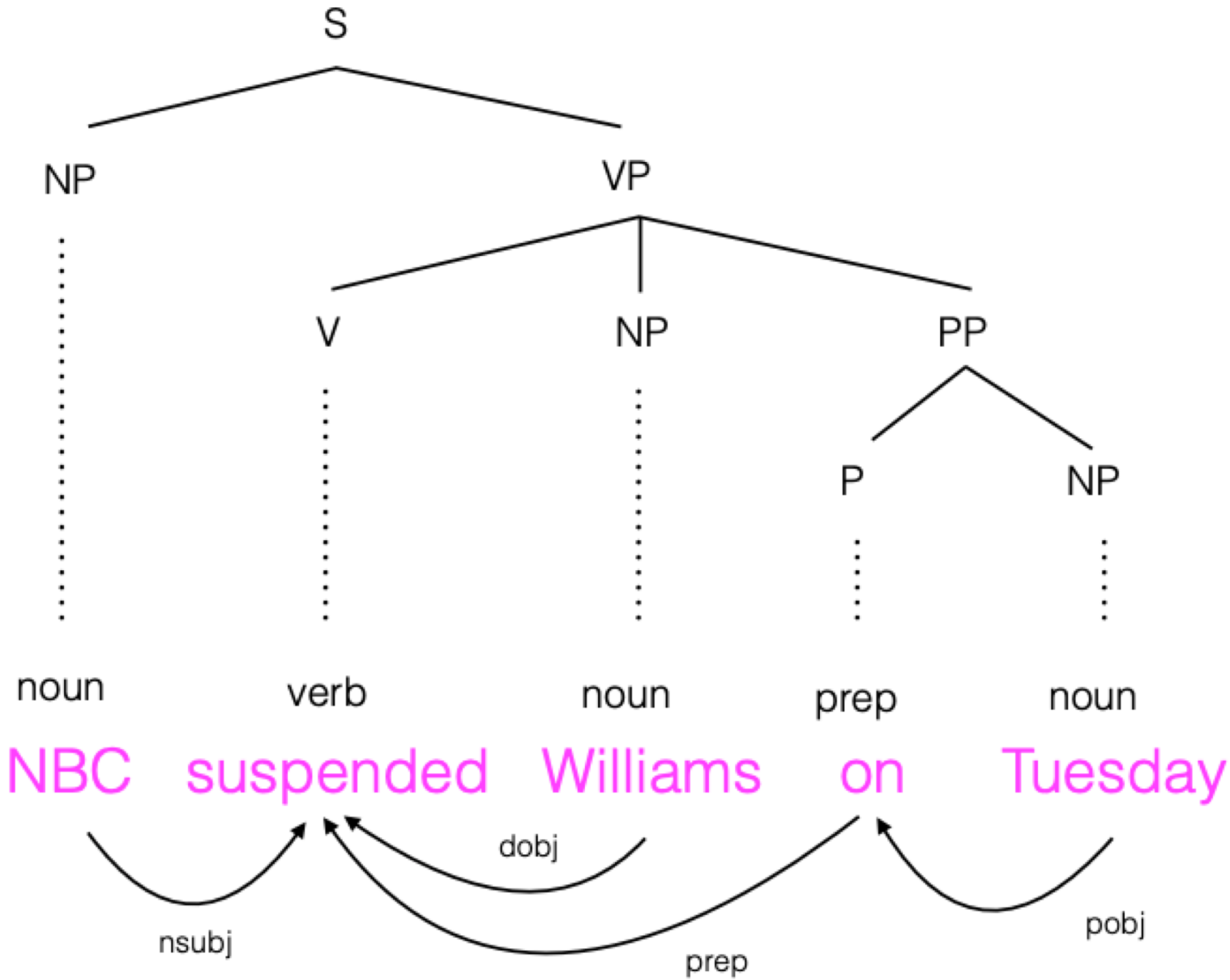

S
NP
VP
V
NP
PP
P
NP
noun
verb
noun
prep
noun
NBC
suspended
Williams
on
Tuesday

Who did what to whom?



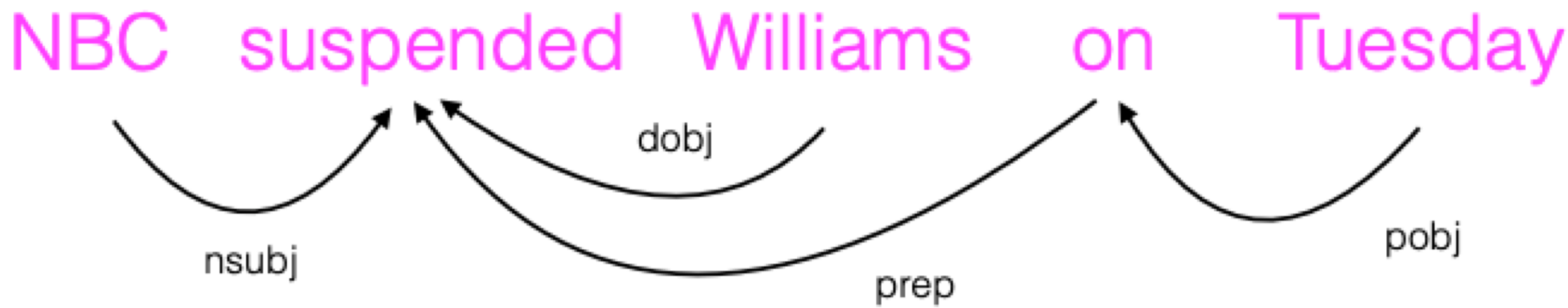
subject: S → NP VP

direct object: S → NP (VP → … NP …)


S
NP
VP
V
NP
PP
P
NP
noun
verb
noun
prep
noun
NBC
suspended
Williams
on
Tuesday
dobj
nsubj
prep
pobj

Captures binary relations between words

- nsubj(NBC, suspended)
- dobj(Williams, suspended)



Data

- NELL SVO triples (604 million nsubj + dobj relations from 230B words on the web)

police	found	five .030 bullets	1
police	found	seven dead rebels	3
police	found	two hidden cameras	2
police	found	wanders lover	1
police	found	211 pounds	4
police	found	Marcia	3
police	found	bank draft	1
police	found	diskette	2
police	found	five marijuana plants	3
police	found	items used	1
police	found	judge	5

Dependency Based Word Embeddings

- Levy & Goldberg, ACL 2014
- http://irsrv2.cs.biu.ac.il:9998/

Target Word	BoW5	BoW2	Deps
batman	nightwing	superman	superman
	aquaman	superboy	superboy
	catwoman	aquaman	supergirl
	superman	catwoman	catwoman
	manhunter	batgirl	aquaman
hogwarts	dumbledore	evernight	sunnydale
	hallows	sunnydale	collinwood
	half-blood	garderobe	calarts
	malfoy	blandings	greendale
	snape	collinwood	millfield
turing	nondeterministic	non-deterministic	pauling
	non-deterministic	finite-state	hotelling
	computability	nondeterministic	heting
	deterministic	buchi	lessing
	finite-state	primality	hamming
florida	gainesville	fla	texas
	fla	alabama	louisiana
	jacksonville	gainesville	georgia
	tampa	tallahassee	california
	lauderdale	texas	carolina
object-oriented	aspect-oriented	aspect-oriented	event-driven
	smalltalk	event-driven	domain-specific
	event-driven	objective-c	rule-based
	prolog	dataflow	data-driven
	domain-specific	4gl	human-centered
dancing	singing	singing	singing
	dance	dance	rapping
	dances	dances	breakdancing
	dancers	breakdancing	miming
	tap-dancing	clowning	busking

Dependency Treebanks

- The major English dependency treebanks converted from the WSJ sections of the PTB (Marcus et al., 1993)
- OntoNotes project (Hovy et al., 2006, Weischedel et al., 2011) adds conversational telephone speech, weblogs, usenet newsgroups, broadcast, and talk shows in English, Chinese and Arabic
- Annotated dependency treebanks created for morphologically rich languages such as Czech, Hindi and Finnish, e.g., Prague Dependency Treebank (Bejcek et al., 2013)
- https://universaldependencies.org/ (122 treebanks, 71 languages)

Parsing Problem

- The parsing problem for a dependency parser is to find the optimal dependency tree $\boldsymbol{y}$ given an input sentence $\boldsymbol{x}$

- assigning a syntactic head i and a label l to every node j corresponding to a word x_j in such a way that the resulting graph is a tree rooted at the node 0

Parsing Problem

- This is equivalent to finding a spanning tree in the complete graph containing all possible arcs

Parsing Algorithms

- Transition based
  - Greedy choice of local transitions guided by a good classifier
  - Deterministic
  - MaltParser (Nivre et al., 2008)
- Graph based
  - Minimum Spanning Tree for a sentence
  - McDonald et al.'s (2005) MSTParser
  - Martins et al.'s (2009) Turbo Parser