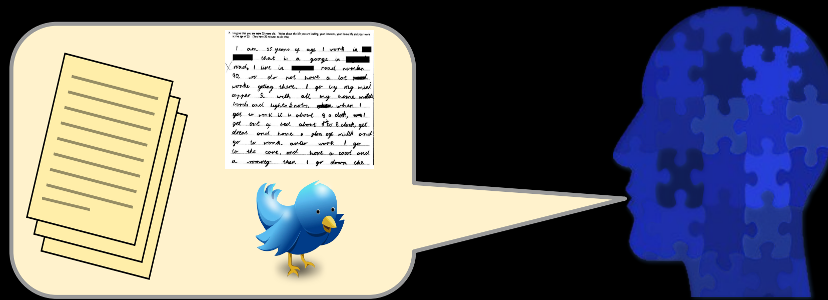
Semantics “Avalanche”:

Word Sense Disambiguation, Dependency Parsing, Semantic Role Labeling/Verb Predicates.

CSE354 - Spring 2020

Natural Language Processing

Tasks



- Word Sense Disambiguation
- Dependency Parsing
- Semantic Role Labeling

how? →

- Traditionally:
  - Probabilistic models
  - Discriminant Learning: e.g. Logistic Regression
  - Transition-Based Parsing
  - Graph-Based Parsing
- Current:
  - Recurrent Neural Network
  - Transformers

GOALS

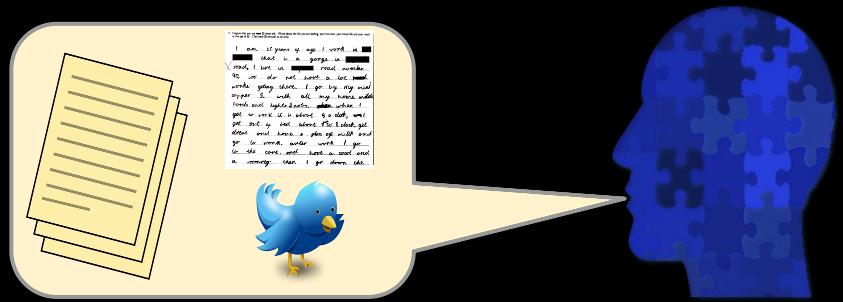
- Define common semantic tasks in NLP.
- Understand linguistic information necessary for semantic processing.
- Learn a couple approaches to semantic tasks.
- Motivate deep learning models necessary to capture language semantics.

- Word Sense Disambiguation
- Dependency Parsing
- Semantic Role Labeling

how? →

- Traditionally:
  - Probabilistic models
  - Discriminant Learning: e.g. Logistic Regression
  - Transition-Based Parsing
  - Graph-Based Parsing
- Current:
  - Recurrent Neural Network
  - Transformers

Tasks

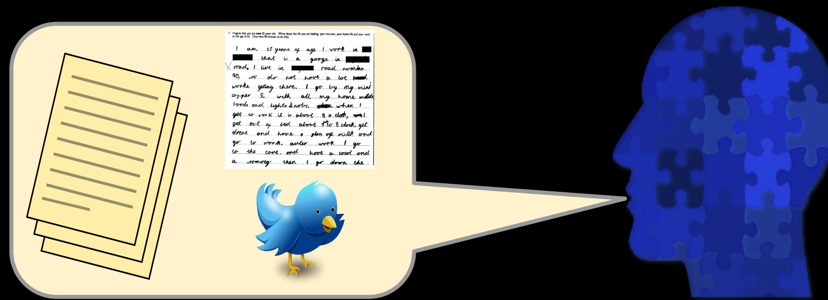


- Word Sense Disambiguation
- Dependency Parsing
- Semantic Role Labeling

how? →

- Traditionally:
  - Probabilistic models
  - Discriminant Learning: e.g. Logistic Regression
  - Transition-Based Parsing
  - Graph-Based Parsing
- Current:
  - Recurrent Neural Network
  - Transformers

Tasks



- **Word Sense Disambiguation**
- Dependency Parsing
- Semantic Role Labeling

how? →

- Traditionally:
  - **Probabilistic models**
  - **Discriminant Learning: e.g. Logistic Regression**
  - Transition-Based Parsing
  - Graph-Based Parsing
- Current:
  - Recurrent Neural Network
  - Transformers

Preliminaries (From SLP, Jurafsky et al., 2013)

Terminology: lemma and wordform

- A **lemma** or **citation form**
  - Same stem, part of speech, rough semantics
- A **wordform**
  - The inflected word as it appears in text

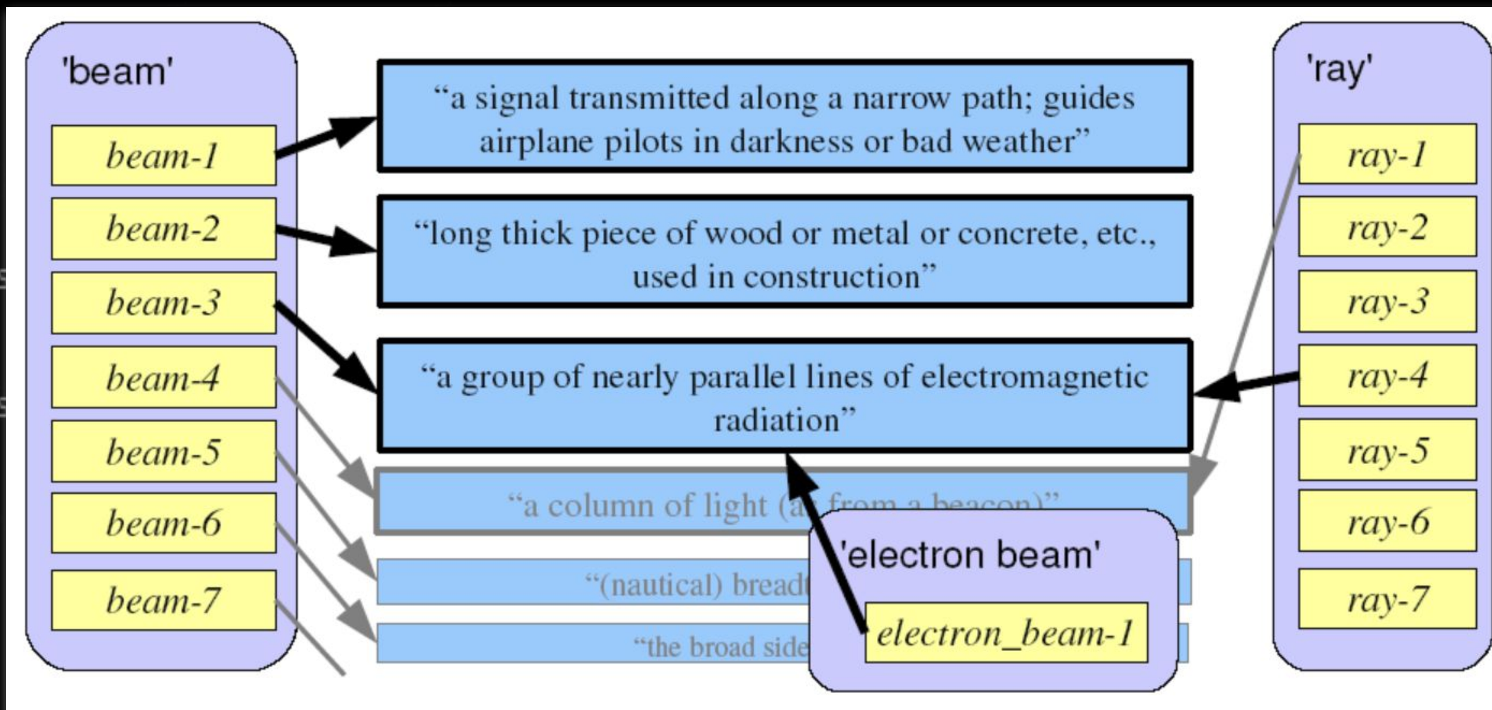
Wordform	Lemma
banks	bank
sung	sing
duermes	dormir

Preliminaries (From SLP, Jurafsky et al., 2013)

Lemmas have senses

- One lemma "bank" can have many meanings:
  - Sense 1: …a **bank** can hold the investments in a custodial account[1]…
  - Sense 2: "…as agriculture burgeons on the east **bank**[2] the river will shrink even more"
- **Sense** (or **word sense**)
  - A discrete representation of an aspect of a word's meaning.
- The lemma **bank** here has two senses

Preliminaries (From SLP, Jurafsky et al., 2013)



- The lemma **bank** here has two senses

Preliminaries (From SLP, Jurafsky et al., 2013)

Homonymy

Homonyms: words that share a form but have unrelated, distinct meanings:

- $bank_1$: financial institution, $bank_2$: sloping land
- bat_1: club for hitting a ball, bat_2: nocturnal flying mammal

1. Homographs (bank/bank, bat/bat)
2. Homophones:
   1. Write and right
   2. Piece and peace

Preliminaries (From SLP, Jurafsky et al., 2013)

Homonymy causes problems for NLP applications

- Information retrieval
  - "bat care"
- Machine Translation
  - bat: murciélago (animal) or bate (for baseball)
- Text-to-Speech
  - bass (stringed instrument) vs. bass (fish)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

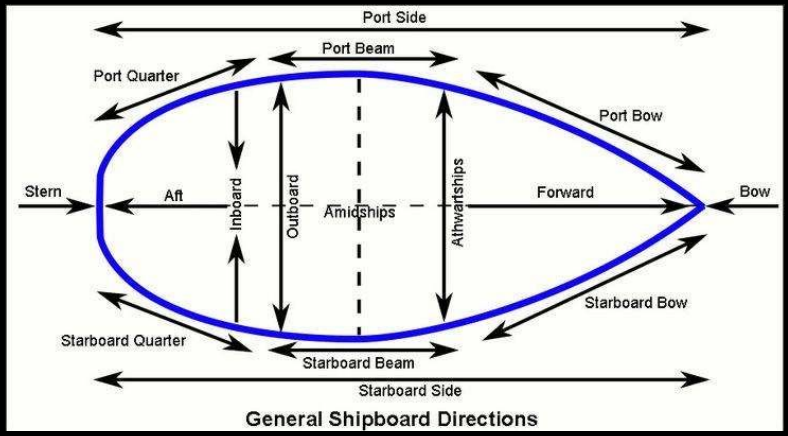
He walked along the **port** next to the steamer.

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

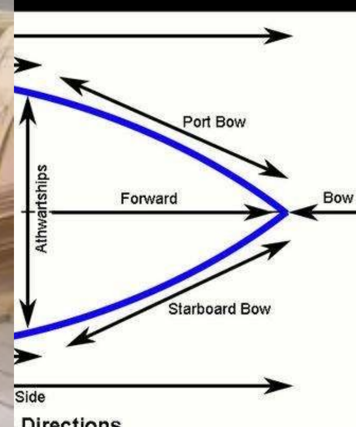


Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.



Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

Word Sense Disambiguation

He put the **port** on the ship.
He walked along the **port** of the steamer.
He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

Word Sense Disambiguation

He put the **port** on the ship.
He walked along the **port** of the steamer.
He walked along the **port** next to the steamer.

As a verb…

1. **port** (put or turn on the left side, of a ship) *"port the helm"*
2. **port** (bring to port) *"the captain ported the ship at night"*
3. **port** (land at or reach a port) *"The ship finally ported"*
4. **port** (turn or go to the port or left side, of a ship) *"The big ship was slowly porting"*
5. **port** (carry, bear, convey, or bring) *"The small canoe could be ported easily"*
6. **port** (carry or hold with both hands diagonally across the body, especially of weapons) *"port a rifle"*
7. **port** (drink port) *"We were porting all in the club after dinner"*
8. **port** (modify (software) for use on a different machine or platform)

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

port.n.1
port.n.2
port.n.3,
port.n.4
port.n.5

Word Sense Disambiguation

A classification problem:

General Form:

f(sent_tokens, (target_index, lemma, POS)) -> word_sense

He walked along the **port** next to the steamer.

Word Sense Disambiguation

A classification problem:

General Form:

f(sent_tokens, (target_index, lemma, POS)) -> word_sense

Logistic Regression (or any discriminative classifier):

$P_{lemma,POS}(\text{sense} = s \mid \text{features})$

He walked along the **port** next to the steamer.

Word Sense Disambiguation

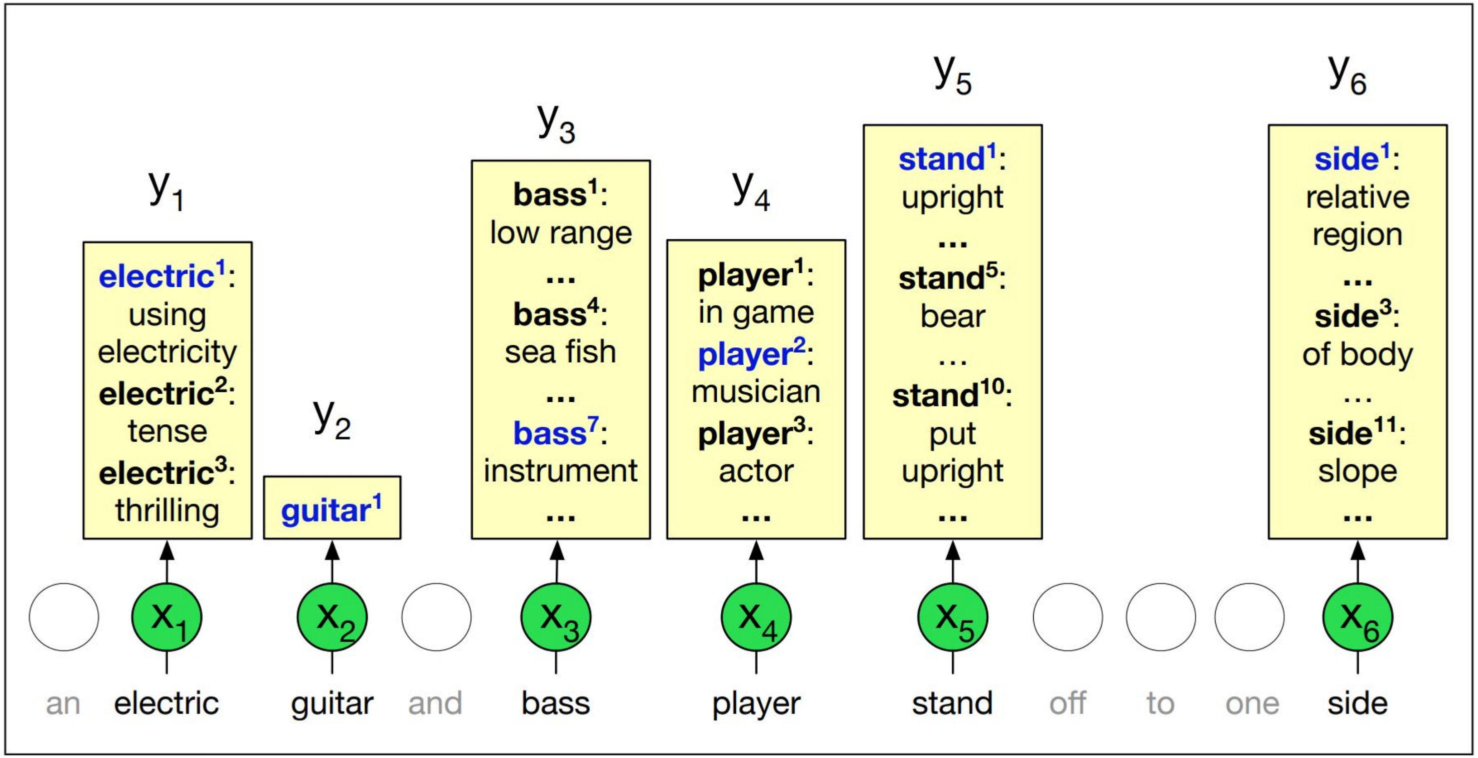


Figure 19.8 The all-words WSD task, mapping from input words (x) to WordNet senses (y). Only nouns, verbs, adjectives, and adverbs are mapped, and note that some words (like *guitar* in the example) only have one sense in WordNet. Figure inspired by Chaplot and Salakhutdinov (2018).

(Jurafsky, SLP 3)

Distributional Hypothesis:

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional Hypothesis:

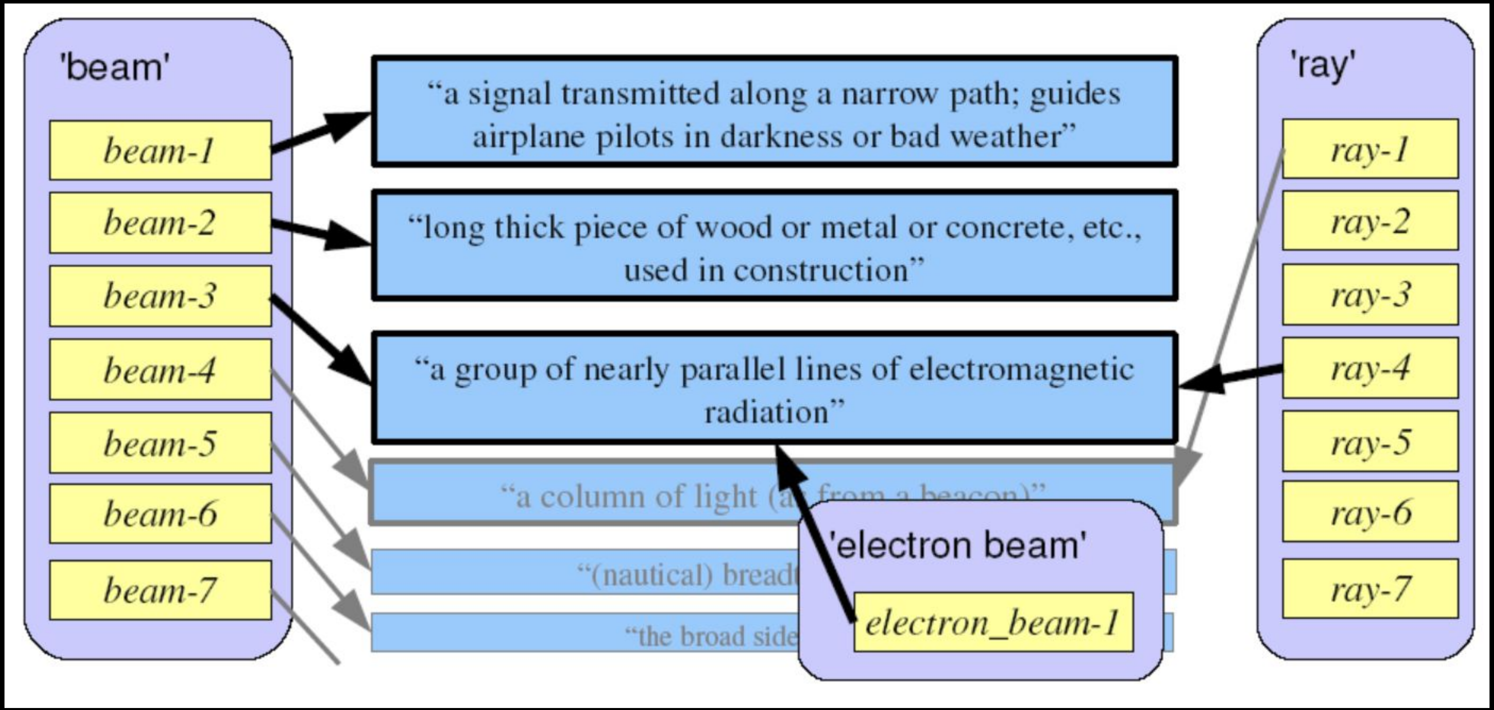
Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional hypothesis -- A word's meaning is defined by all the different contexts it appears in (i.e. how it is "distributed" in natural language).

Firth, 1957: "*You shall know a word by the company it keeps*"

The nail hit the beam behind the wall.

Distributional Hypothesis



The nail hit the beam behind the wall.

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. Lesk algorithm
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).

1 and 2 Mirror POS Tagging:

Features to represent words in the exact context

Improvements:

- use *lemmas* rather than unique words (be, was, is, were => "be")
- Use POS of surrounding words as well.

He addressed the strikers at the rally.

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. **Lesk algorithm**
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Lesk Algorithm for WSD

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ← set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
          max-overlap ← overlap
          best-sense ← sense
  end
  return(best-sense)
```

Figure 19.10 The Simplified Lesk algorithm. The COMPUTEOVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the *context* in a more complex way.

Lesk Algorithm for WSD

- bank.n.1 (sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
- bank.n.2 (a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"

```
  overlap ← COMPUTEOVERLAP(signature, context)
  if overlap > max-overlap then
      max-overlap ← overlap
      best-sense ← sense
end
return(best-sense)
```

The bank can guarantee deposits will cover future tuition costs, ...

- bank.n.1 (sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
- bank.n.2 (a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"
- ...
- bank.n.4 (an arrangement of similar objects in a row or in tiers) "he operated a bank of switches"
- ...
- bank.n.8 (a building in which the business of banking transacted) "the bank is on the corner of Nassau and Witherspoon"
- bank.n.9 (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) "the plane went into a steep bank"

end
return(*best-sense*)

The bank can guarantee deposits will cover future tuition costs, ...

Lesk Algorithm for WSD

- **striker.n.1** (a forward on a soccer team)
- **striker.n.2** (someone receiving intensive training for a naval technical rating)
- **striker.n.3** (an employee on strike against an employer)
- **striker.n.4** (someone who hits) *"a hard hitter"; "a fine striker of the ball"; "blacksmiths are good hitters"*
- **striker.n.5** (the part of a mechanical device that strikes something)

overlap ← COMPUTEOVERLAP(*signature*, *context*)
if *overlap* > *max-overlap* **then**
max-overlap ← *overlap*
best-sense ← *sense*
end
return(*best-sense*)

He addressed the strikers at the rally.

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. **Lesk algorithm**
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. Lesk algorithm
   E.g. compare context to sense definitions.
4. **Selectors -- other *target words* that appear with same context** *E.g. counts for any selector.*
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Selectors

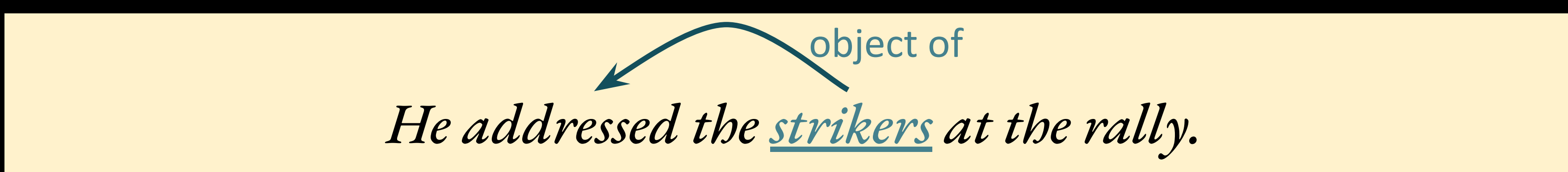
… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse

Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse



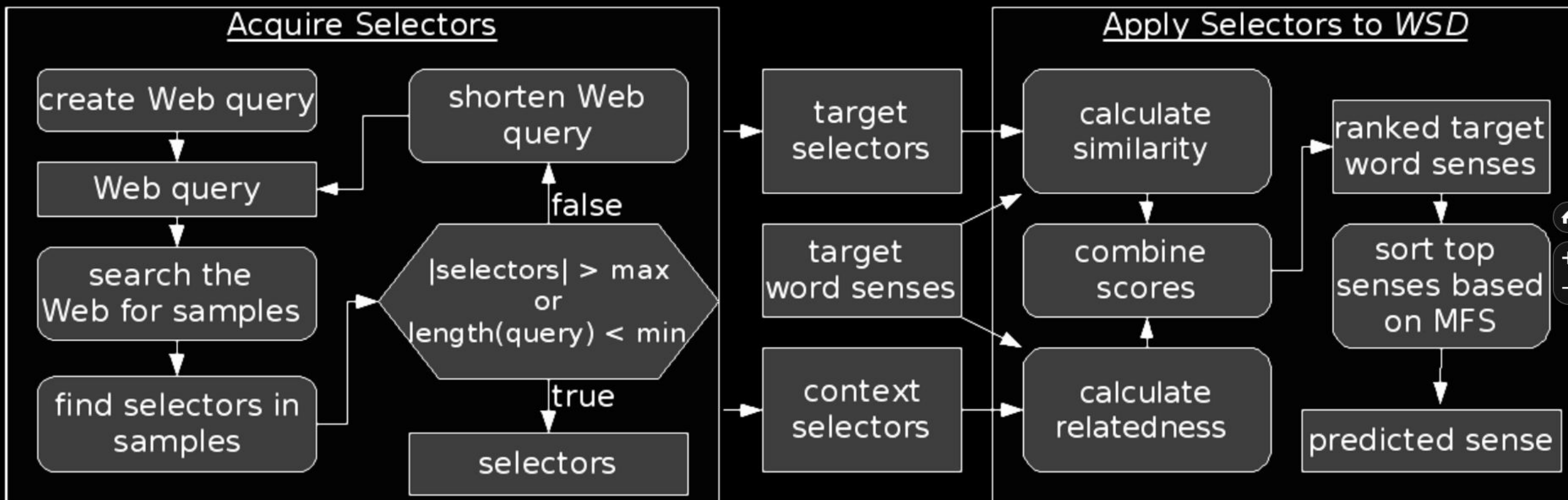
Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse (Lin, 1997)

Web version: Local context defined by lexical patterns matched on the Web (Schwartz, 2008).

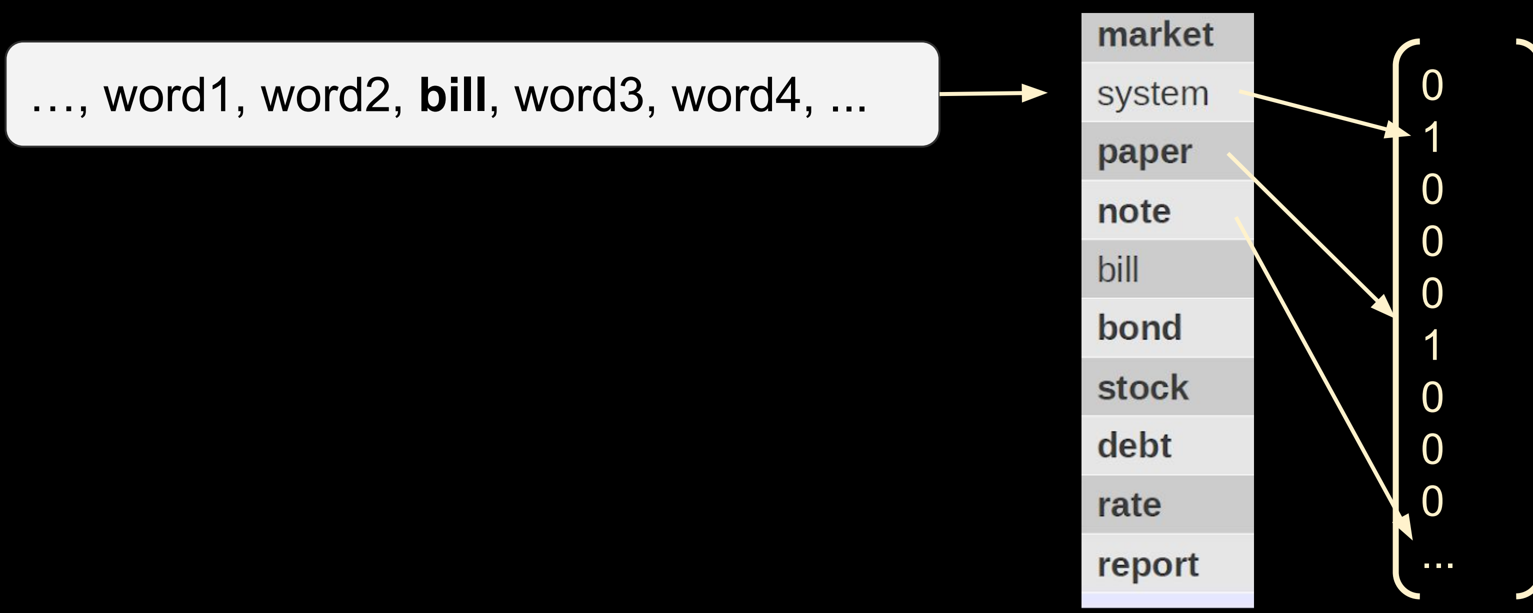
*"He addressed the * at the rally."*


Acquire Selectors
create Web query
Web query
search the Web for samples
find selectors in samples
shorten Web query
false
|selectors| > max or length(query) < min
true
selectors
target selectors
target word senses
context selectors
Apply Selectors to WSD
calculate similarity
combine scores
calculate relatedness
ranked target word senses
sort top senses based on MFS
predicted sense

Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

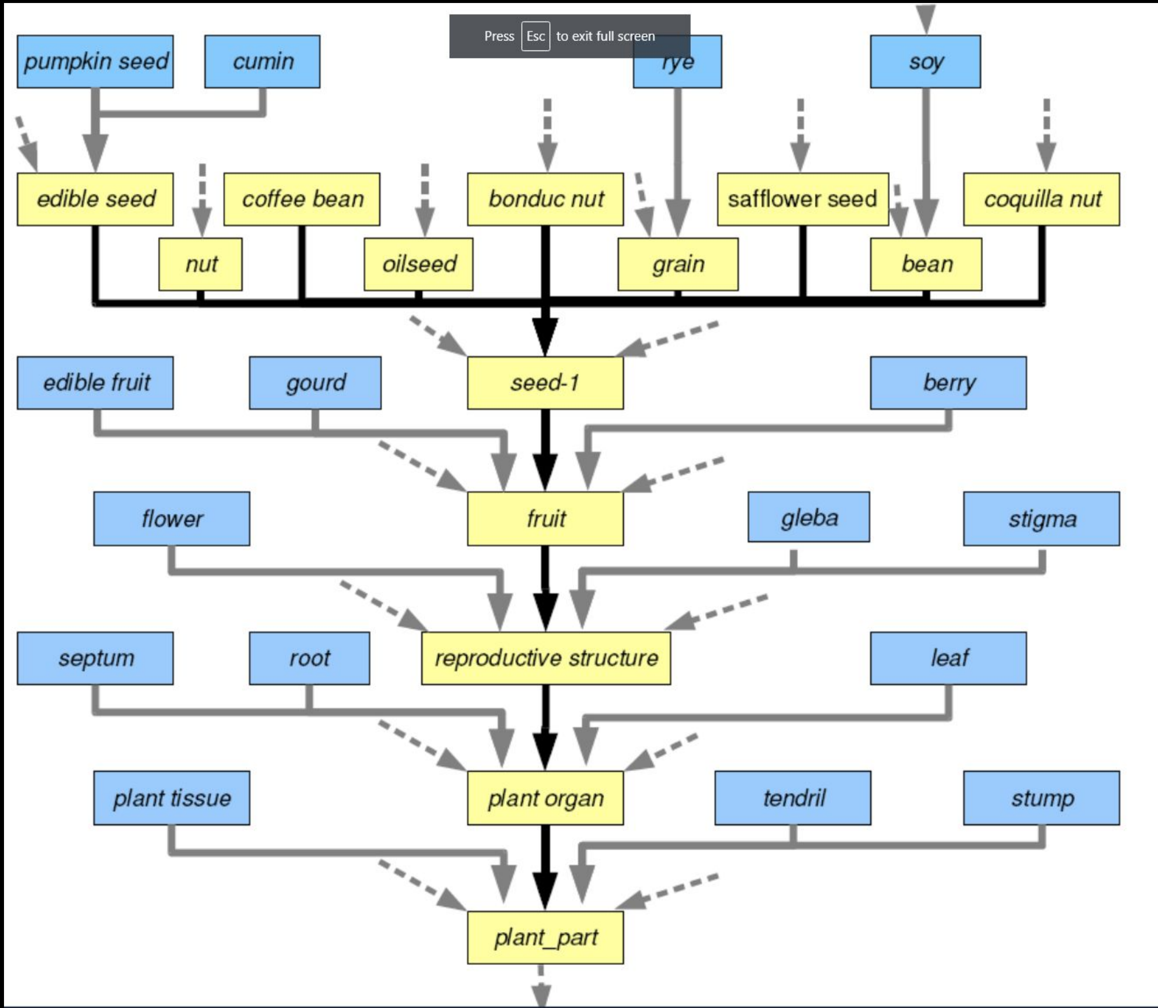
"…, but the bill now under discussion"



Selectors

Leverages *hypernymy:*

concept1 <is-a> concept2



Selectors

"He addressed the strikers at the rally."

he	addressed	strikers	rally
man	scolded	crowd	protest
owners	rallyied	students	demonstration
Mary	kept	workers	work
...	...	audience	stadium
		supporters	...
		...	

Why Are Selectors Effective?

Sets of selectors tend to vary extensively by word sense:

bill-n.1	*bill-n.2*	*bill-n.3*
bill	bill	**market**
it	**staff**	system
legislation	system	**paper**
system	**money**	**note**
program	**time**	bill
law	it	**bond**
plan	**tax**	**stock**
you	**work**	**debt**
measure	**rent**	**rate**
project	**tuition**	**report**

occur-v.1	*occur-v.2*	*occur-v.3*
be	go	go
happen	get	**look**
occur	Come	**break**
go	have	**remove**
take	**try**	**find**
work	**lead**	get
come	**listen**	**place**
see	work	**keep**
have	be	**stick**
change	**belong**	**stop**

- *Polls show wide, generalized support for some vague concept of service, but the* ***bill*** *now under discussion lacks any passionate public backing.*
  training set never contained: “but the _ now under”
- *… in his lecture, refers to the “startling experience which almost every person confesses, that particular passages of conversation and action have* ***occurred*** *to him in the same order before, whether dreaming or waking …*
  small context is contradictory:
    “action have occurred” => occur-v.1 (“to happen or take place”)
    “occurred to him” => occur-v.2 (“to come to mind”)

bill-n.1	***bill-n.2***	***bill-n.3***
bill	bill	**market**
it	**staff**	system
legislation	system	**paper**
system	**money**	**note**
program	**time**	bill
law	it	**bond**
plan	**tax**	**stock**
you	**work**	**debt**
measure	**rent**	**rate**
project	**tuition**	**report**

occur-v.1	***occur-v.2***	***occur-v.3***
be	go	go
happen	get	**look**
occur	Come	**break**
go	have	**remove**
take	**try**	**find**
work	**lead**	get
come	**listen**	**place**
see	work	**keep**
have	be	**stick**
change	**belong**	**stop**

Supervised Selectors

	base	**w/ sels**	***mfs***	***tests***
noun	87.9	**91.7**	*80.9*	*2559*
verb	83.3	**83.7**	*76.5*	*2292*
both	85.7	**87.9**	*78.8*	*4851*

Accuracy over SemEval-2007: Task 17.

Supervised Selectors

	base	**w/ sels**	***mfs***	***tests***
noun	87.9	**91.7**	*80.9*	*2559*
verb	83.3	**83.7**	*76.5*	*2292*
both	85.7	**87.9**	*78.8*	*4851*

Accuracy over SemEval-2007: Task 17.

	base	**w/ sels**	***mfs***	***tests***
noun	68.5	**72.1**	*54.1*	*1766*
verb	72.0	72.4	*57.9*	*1927*
adjective	49.4	**53.4**	*54.7*	*148*
all	69.4	**71.5**	*56.1*	*3841*

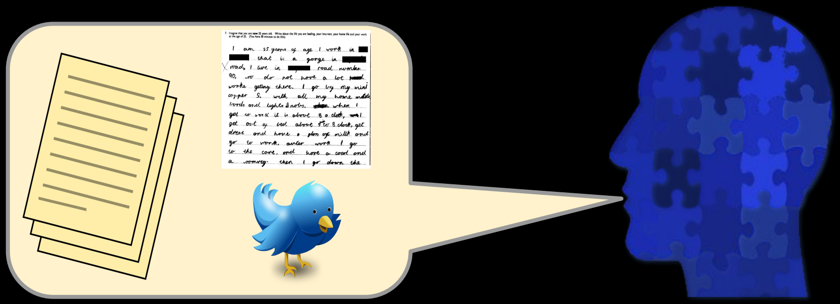
Accuracy over seneval-3 Lexical Sample.
(fine-grained senses compared to SemEval)

More Background on WSD

https://prezi.com/m86pd1zbe_fy/?utm_campaign=share&utm_medium=copy

Covers a few approaches plus more background on “lexical semantics” in general.

Tasks

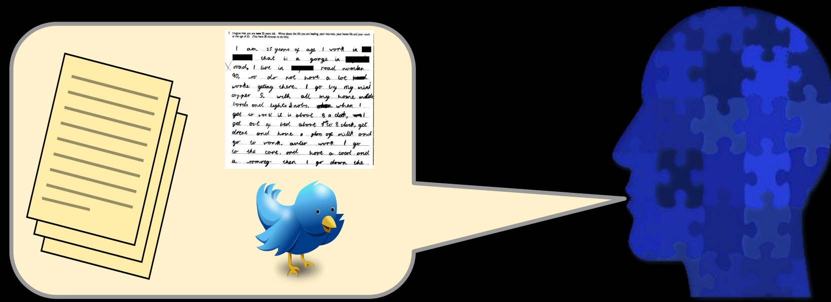


- **Word Sense Disambiguation**
- Dependency Parsing
- Semantic Role Labeling

how?

- Traditionally:
  - **Probabilistic models**
  - **Discriminant Learning: e.g. Logistic Regression**
  - Transition-Based Parsing
  - Graph-Based Parsing
- Current:
  Recurrent Neural Network

Tasks

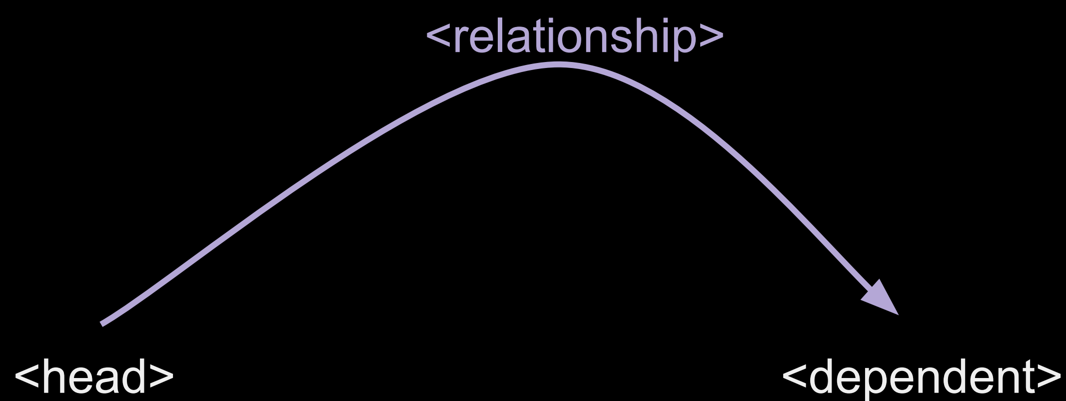


- Word Sense Disambiguation
- **Dependency Parsing**
- **Semantic Role Labeling**

how?

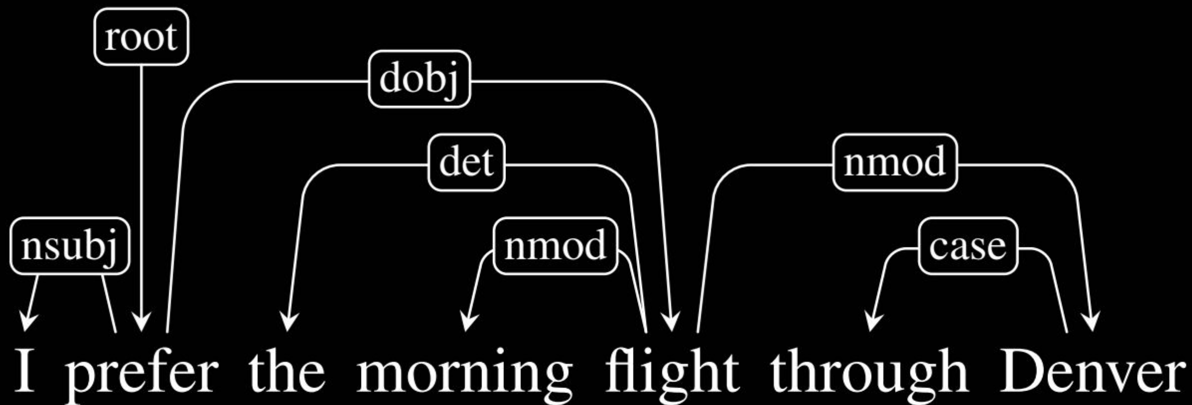
- Traditionally:
  - **Probabilistic models**
  - Discriminant Learning: e.g. Logistic Regression
  - **Transition-Based Parsing**
  - **Graph-Based Parsing**
- Current:
  Recurrent Neural Network

Dependency Parsing



dependency -- binary asymmetrical relation between tokens

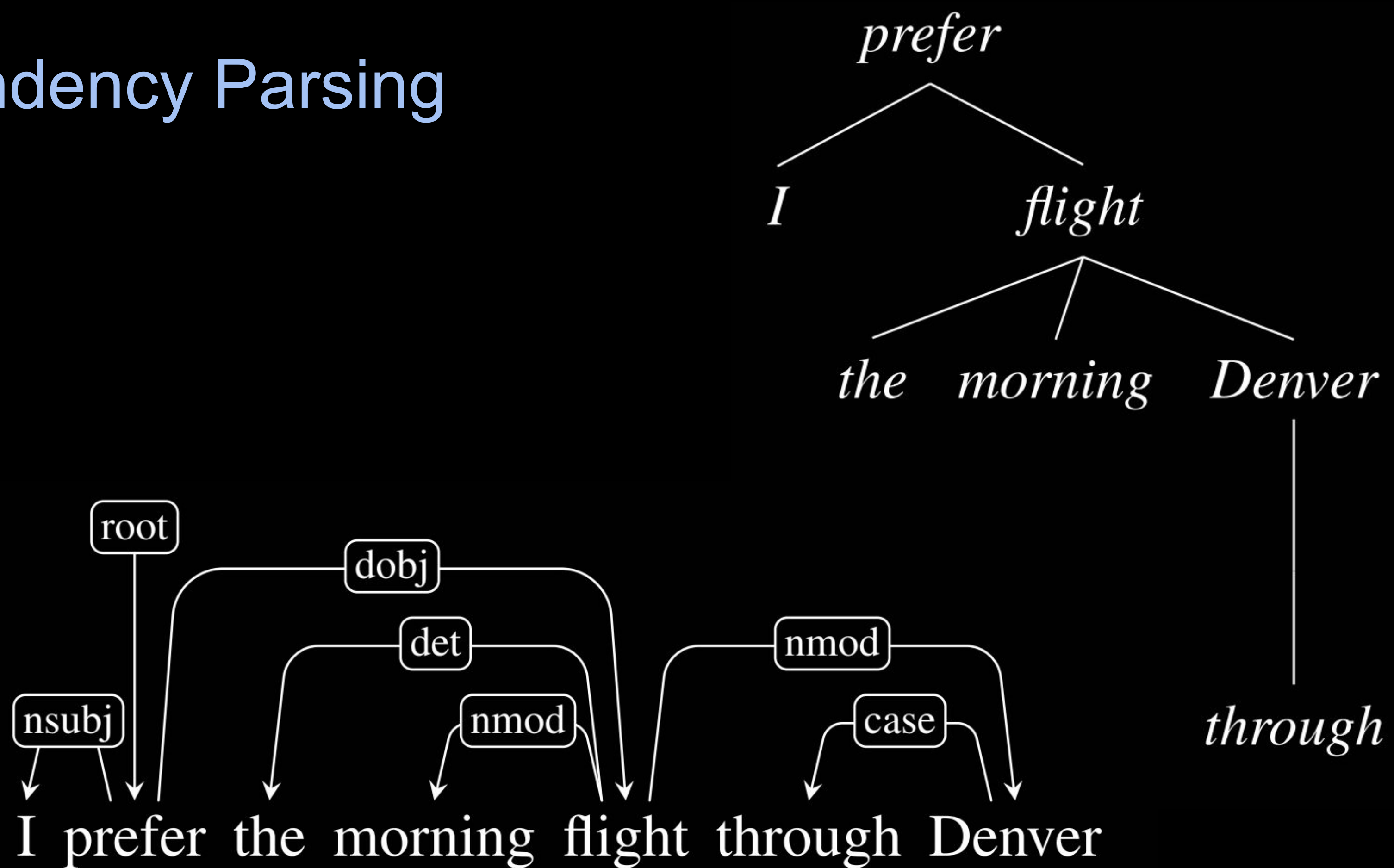
Dependency Parsing



(13.1)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing



(13.1)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing

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

Figure 13.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing

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*.

Figure 13.3 Examples of core Universal Dependency relations.

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing

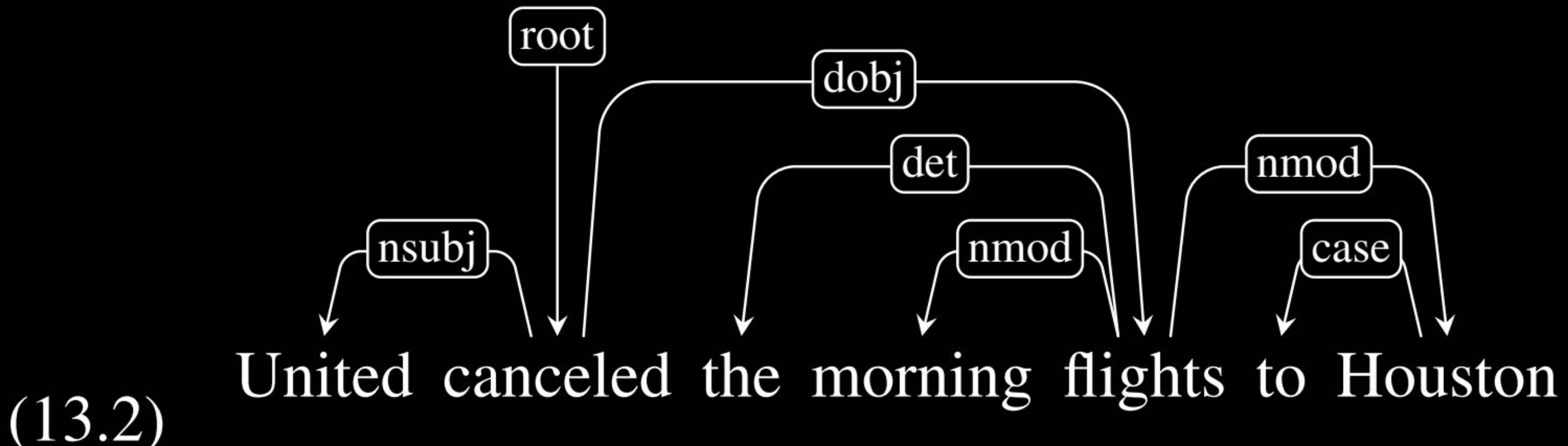
Verbal Predicate -- like a function, takes arguments: "United" and "the flight" in this case.

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*.

Figure 13.3 Examples of core Universal Dependency relations.

(From SLP 3rd ed., Jurafsky and Martin 2018)

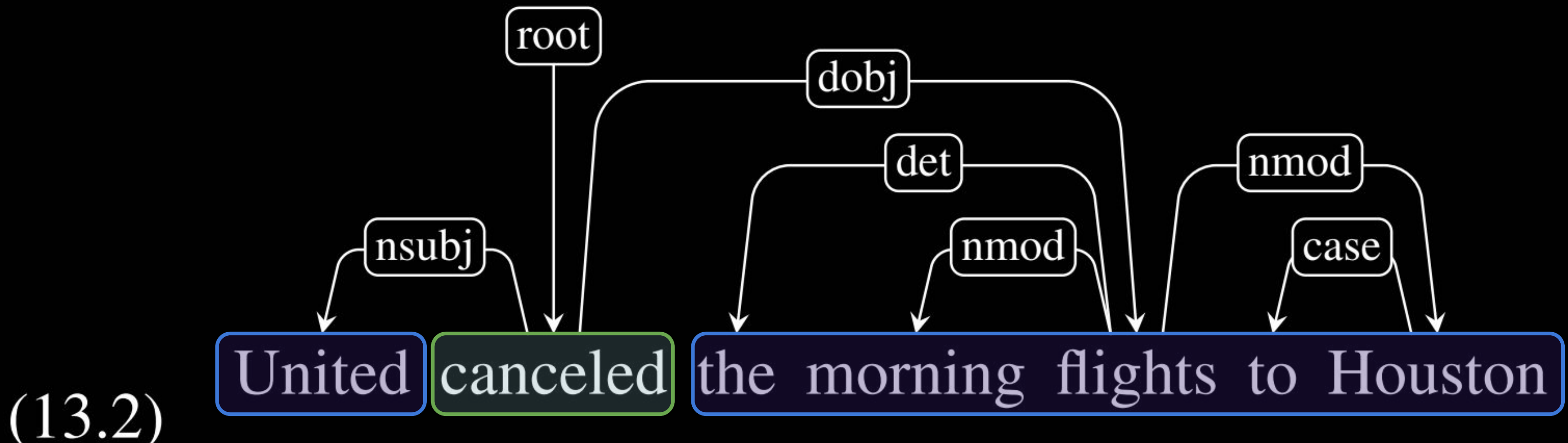
Dependency Parsing -- Verbal Predicates



(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- Verbal Predicates

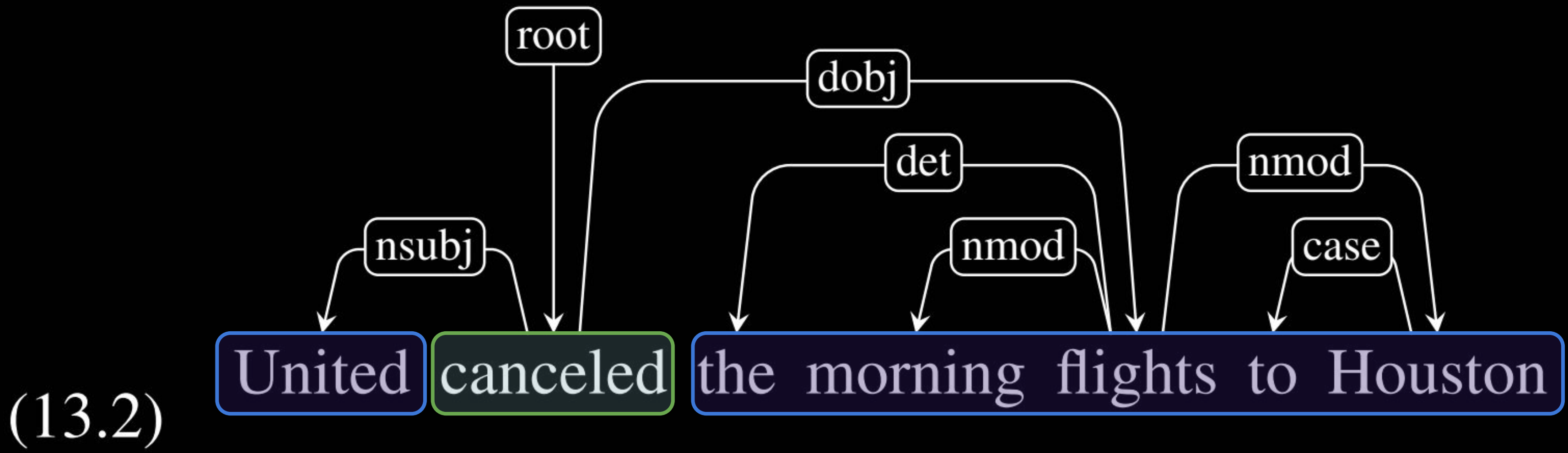
cancel("United", "the morning flights to Houston")



(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- Verbal Predicates

to_call_off(“United”, “the morning flights to Houston”)

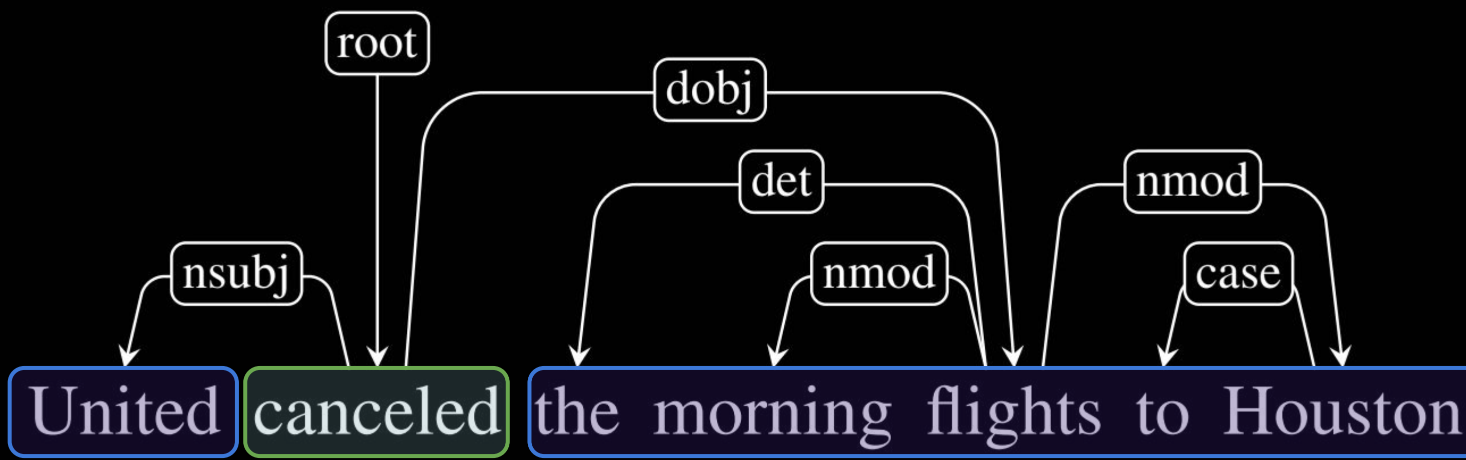


(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- Verbal Predicates
Semantic Roles

to_call_off(agent="United", event="the morning flights to Houston")

(13.2)



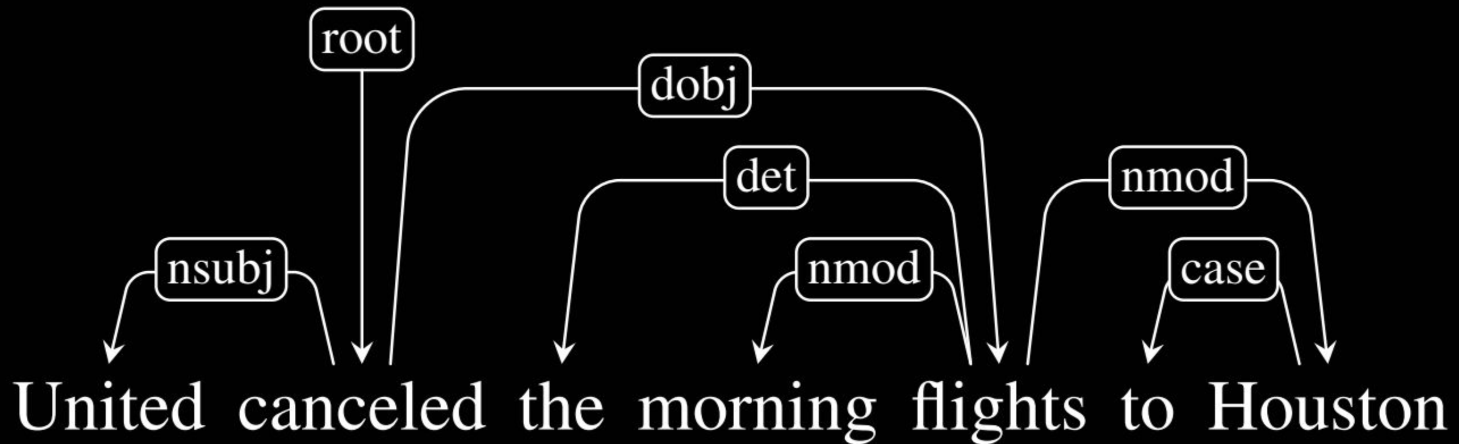
(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- How to Represent?

A Graph: G = [(V1, A1), (V1, A2), …]   (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex



(13.2)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Transition-based Dependency Parsing

Inspired by “Shift-reduce parsing” -- process one word at a time, using a stack to keep some sort of memory.

Elements:

- *S*: stack, initialized with “ROOT”
- *B*: input buffer, initialized with tokens (w1, w2, ….) of sentence
- *A:* set of dependency arcs, initialized empty
- *T:* Actions, given *wi* (next token in stack)

Transition-based Dependency Parsing

Inspired by "Shift-reduce parsing" -- process one word at a time, using a stack to keep some sort of memory.

Elements:

- *S*: stack, initialized with "ROOT"
- *B*: input buffer, initialized with tokens (w1, w2, ….) of sentence
- *a:* set of dependency arcs, initialized empty
- Actions, given *wi* (next token in stack)
    - *shift*(*B*,*S*): move w from *B* to *S*
    - *left-arc(S,A):* make top of stack **head** of next item: add to A; remove dependent from stack
    - *right-arc(S,A):* make top of stack **dependent** of next item: add to A; remove dep from stack

Using discriminative classifiers (i.e. logistic regression) to make decisions.

Transition-based Dependency Parsing

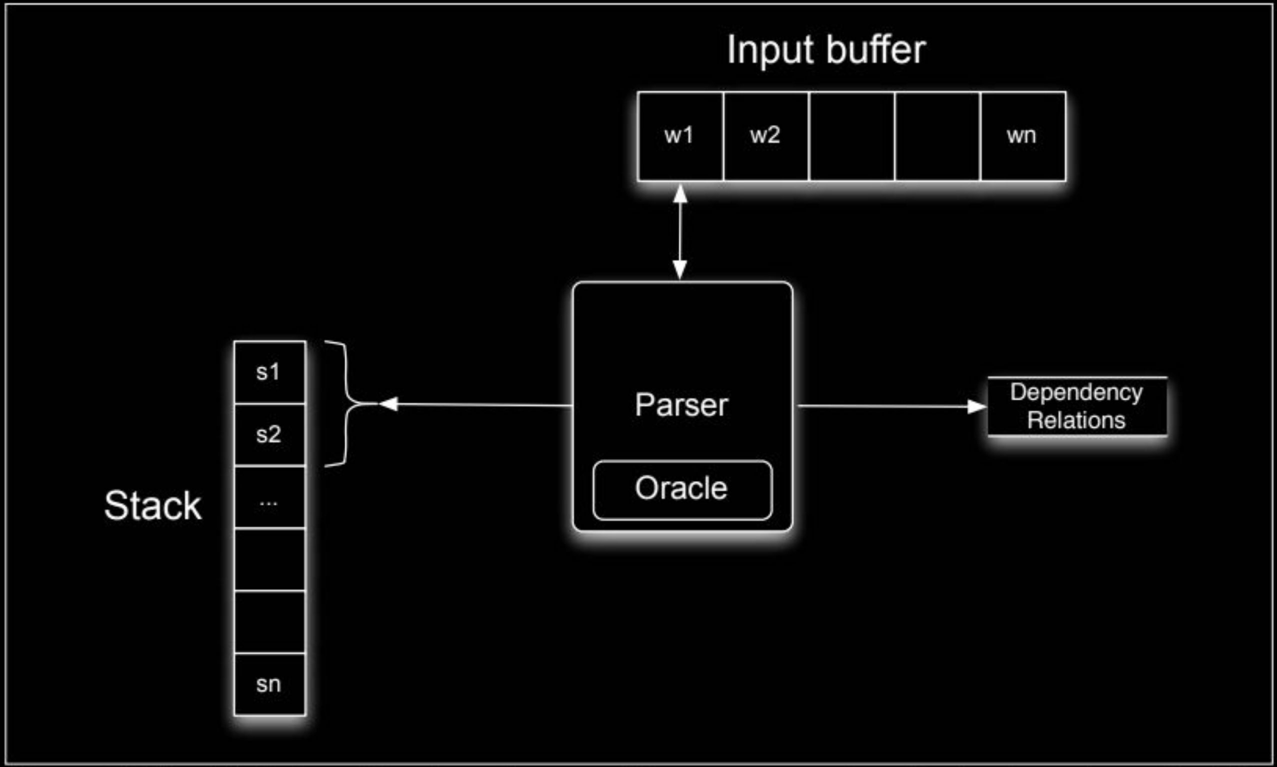


Figure 13.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

(From SLP 3rd ed., Jurafsky and Martin 2018)

Transition-based Dependency Parsing

function DEPENDENCYPARSE(*words*) **returns** dependency tree

state ← {[root], [*words*], [] } ; initial configuration
while *state* **not final**
  t ← ORACLE(*state*) ; choose a transition operator to apply
  state ← APPLY(*t*, *state*) ; apply it, creating a new state
return *state*

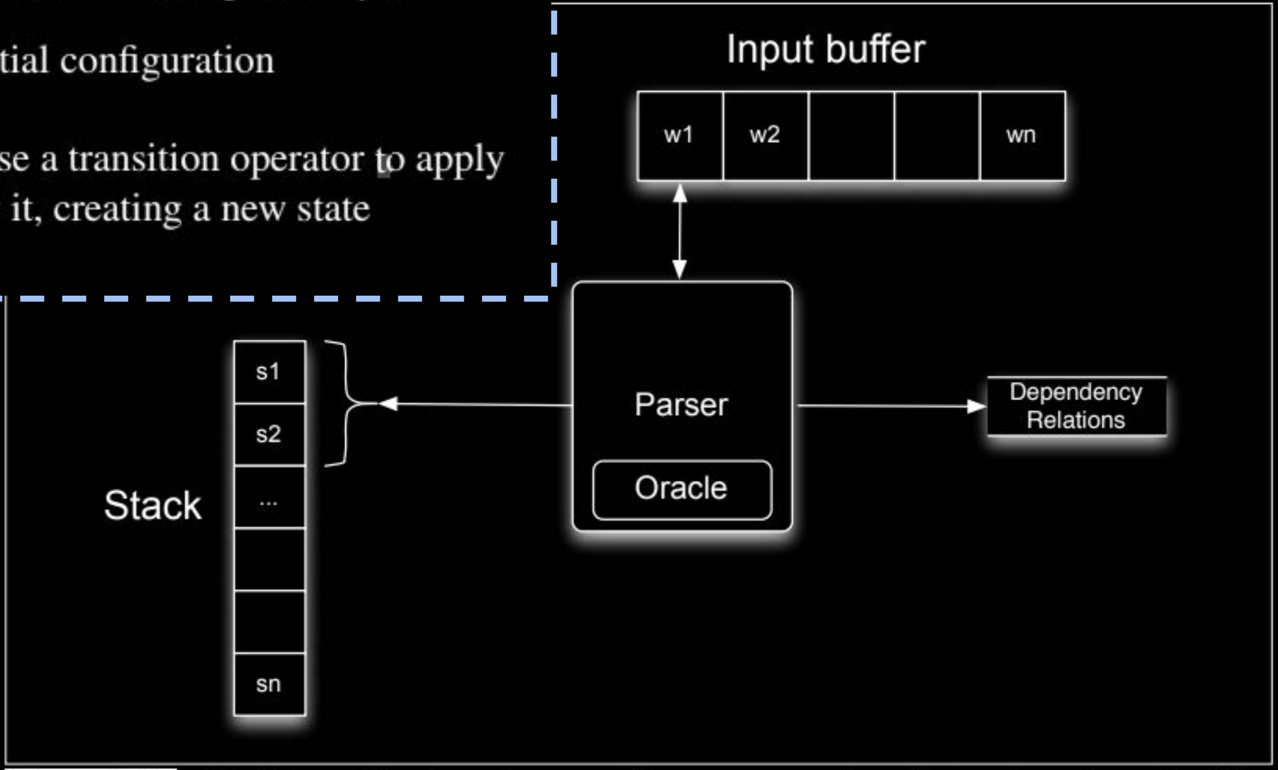


Figure 13.5 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action based on consulting an oracle that examines the current configuration.

(From SLP 3rd ed., Jurafsky and Martin 2018)

Transition-based Dependency Parsing

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)          ; choose a transition operator to apply
    state ← APPLY(t, state)  ; apply it, creating a new state
  return state
```

(13.5) Book me the morning flight

Let's consider the state of the configuration at Step 2, after the word *me* has been pushed onto the stack.

Stack	Word List	Relations
[root, book, me]	[the, morning, flight]	

The correct operator to apply here is RIGHTARC which assigns *book* as the head of *me* and pops *me* from the stack resulting in the following configuration.

Stack	Word List	Relations
[root, book]	[the, morning, flight]	(book $\rightarrow$ me)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Transition-based Dependency Parsing

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Figure 13.7 Trace of a transition-based parse.

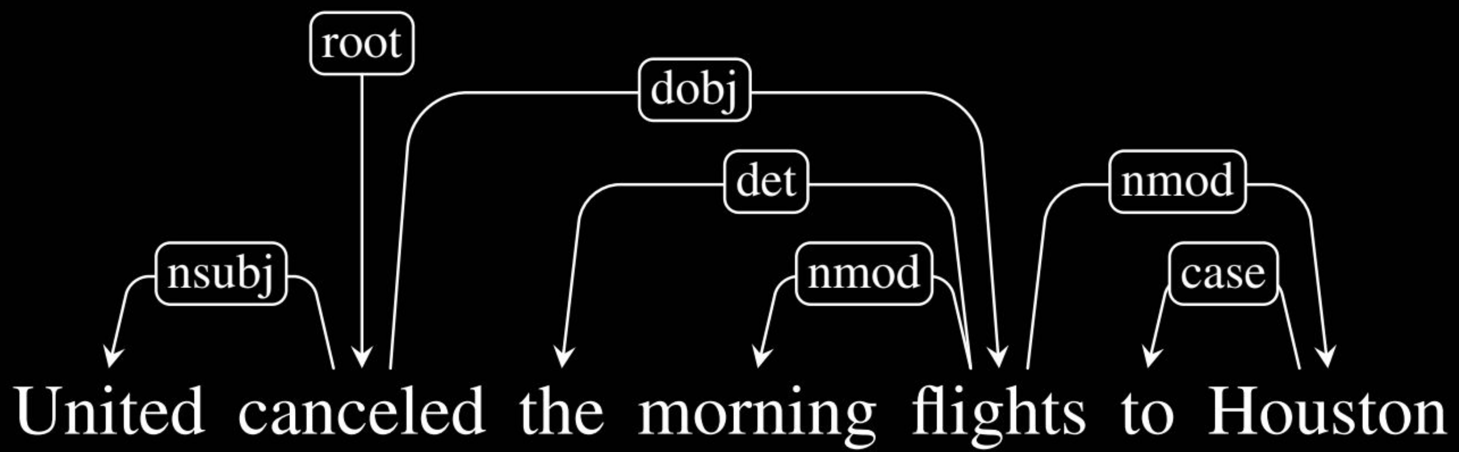
(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- How to Represent?

A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex



(13.2)

(From SLP 3rd ed., Jurafsky and Martin 2018)

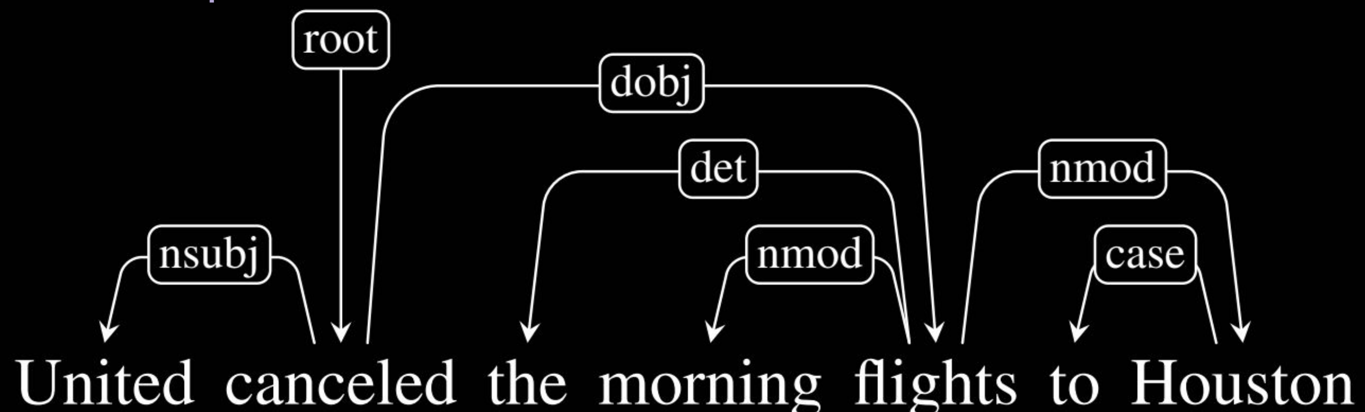
Dependency Parsing -- How to Represent?

A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

Projectivity: Given head, dependent; for every word between head and dependent there exists a path from head to that word



(13.2)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- How to Represent?

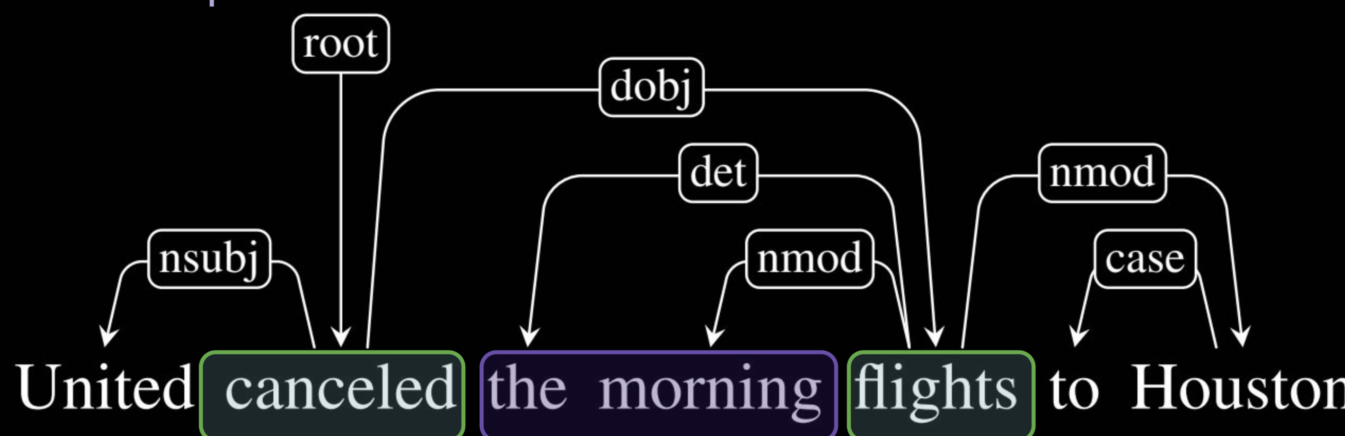
A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

Projectivity: Given head, dependent; for every word between head and dependent there exists a path from head to that word

(13.2)



(From SLP 3rd ed., Jurafsky and Martin 2018)

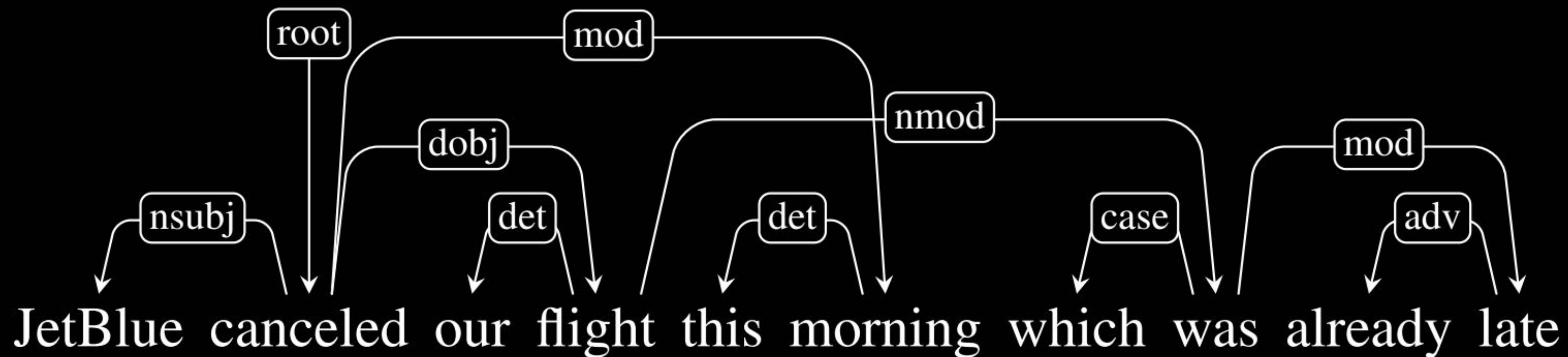
Dependency Parsing -- How to Represent?

A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

Projectivity: Given head, dependent; for every word between head and dependent there exists a path from head to that word



(13.3)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- How to Represent?

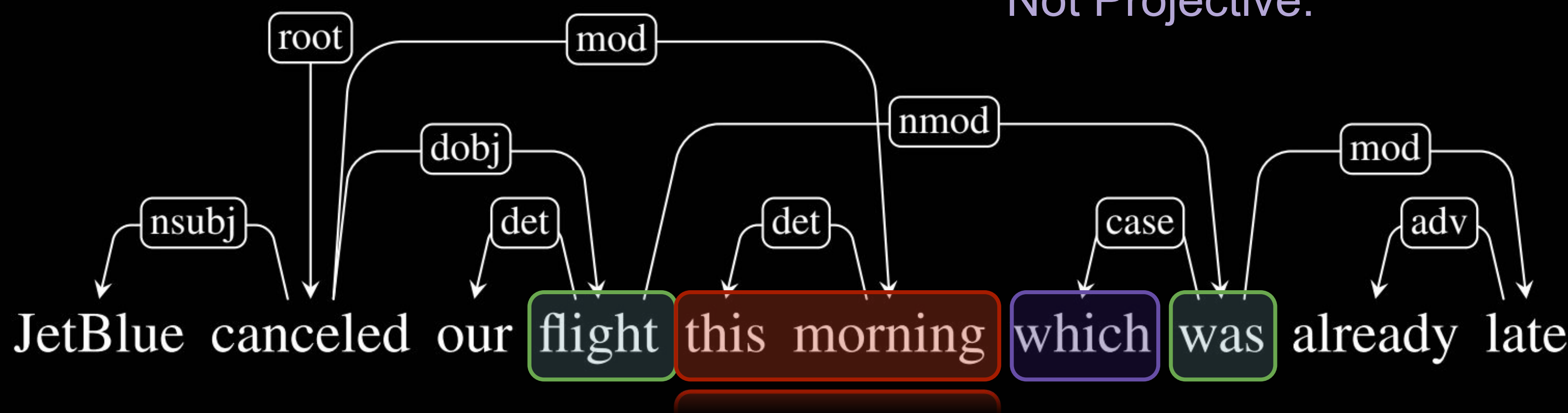
A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

Projectivity: Given head, dependent; for every word between head and dependent there exists a path from head to that word.

Not Projective:



(13.3)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Dependency Parsing -- How to Represent?

A Graph: G = [(V1, A1), (V1, A2), …]   (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

Projectivity: Given head, dependent; for every word between head and dependent there exists a path from head to that word.

Not Projective:

Why do we care? Dependency trees from Context-Free Grammars are guaranteed to be projective; Thus, transition based techniques are certain to have errors occasionally on non-projective dependency graphs.

(From SLP 3rd ed., Jurafsky and Martin 2018)

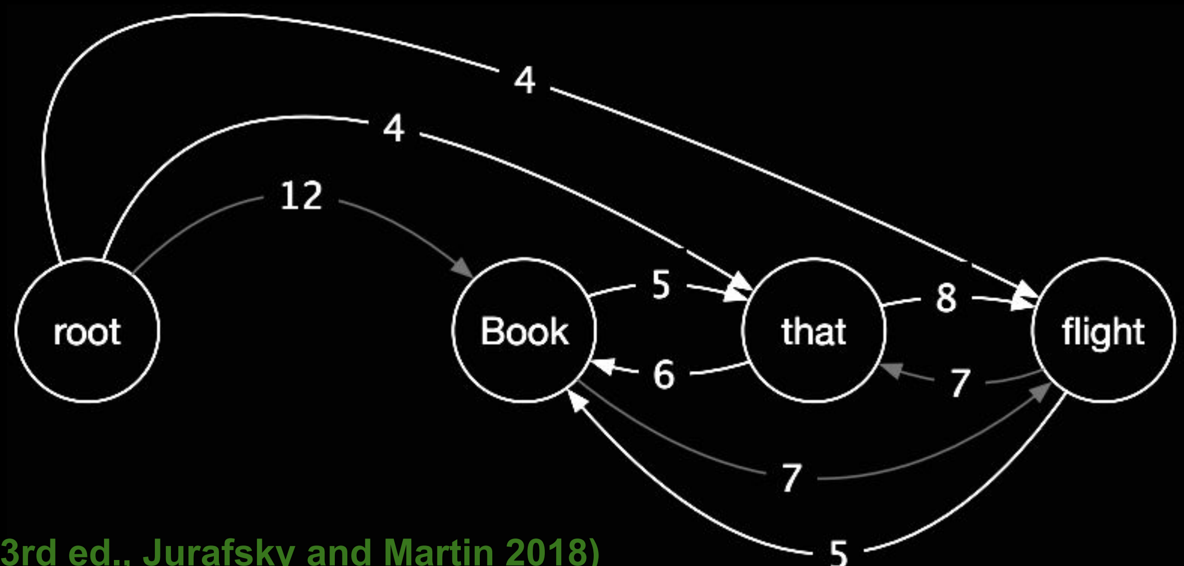
Graph-based Approaches

A Graph: G = [(V1, A1), (V1, A2), …] (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

General Idea: Search through all possible trees and pick best.



(From SLP 3rd ed., Jurafsky and Martin 2018)

Graph-based Approaches

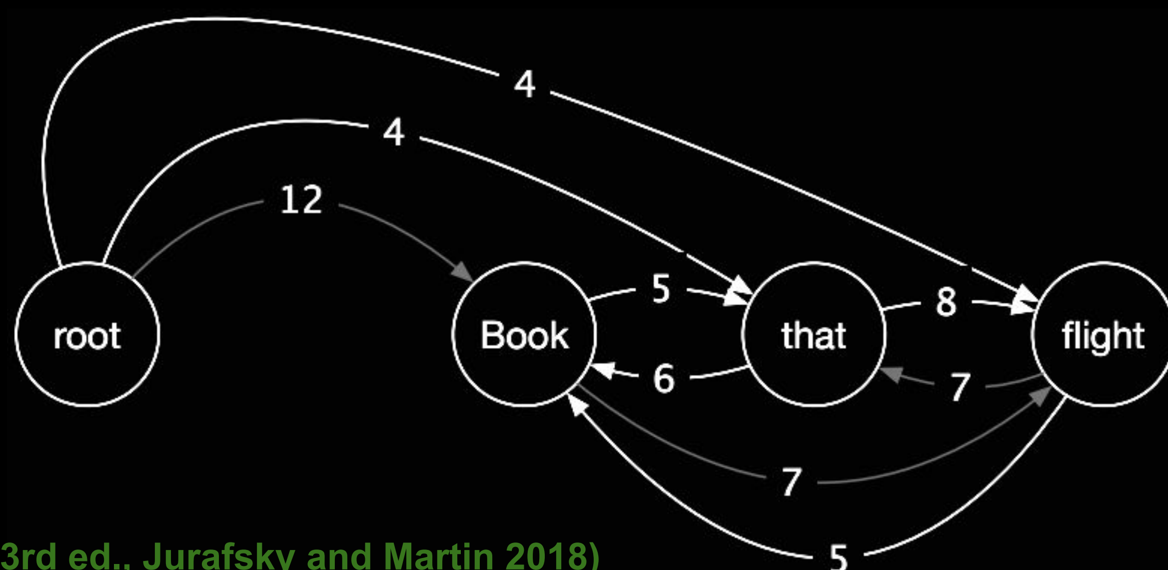
A Graph: G = [(V1, A1), (V1, A2), …]   (vertices and arcs)

Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

General Idea: Search through all possible trees and pick best.

General approach: For each word, pick the most likely head. Then check if still a fully-connected tree, and adjust.



(From SLP 3rd ed., Jurafsky and Martin 2018)

Graph-based Approaches

A Graph: G = [(V1, A1), (V1, A2), …]   (vertices and arcs)

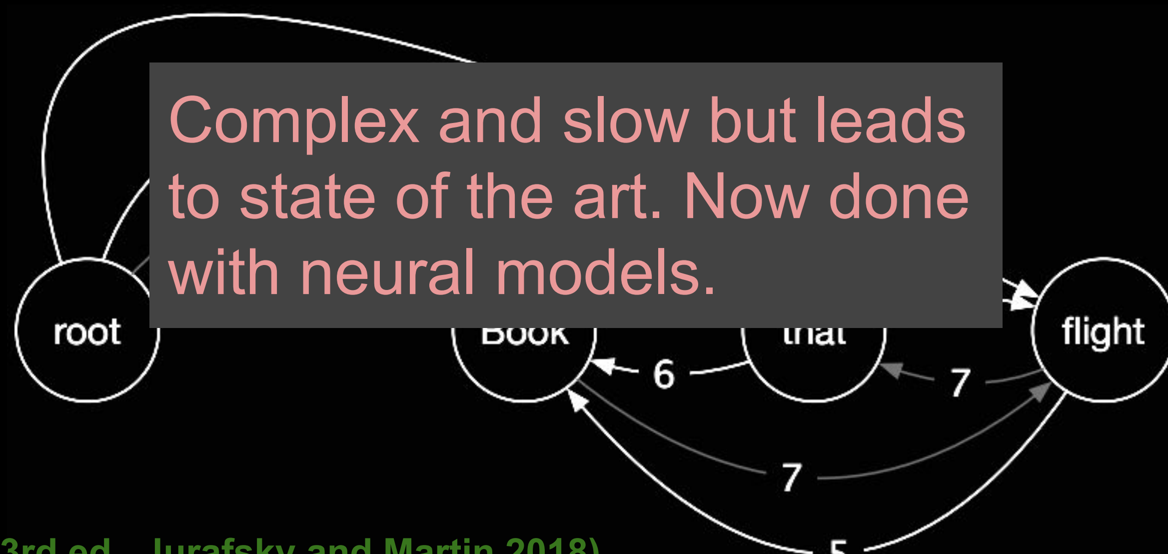
Restrictions:

1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governer); i.e. only one incoming arc
3) unique path from ROOT to every vertex

General Idea: Search through all possible trees and pick best.

General approach: For each word, pick the most likely head. Then check if still a fully-connected tree, and adjust.

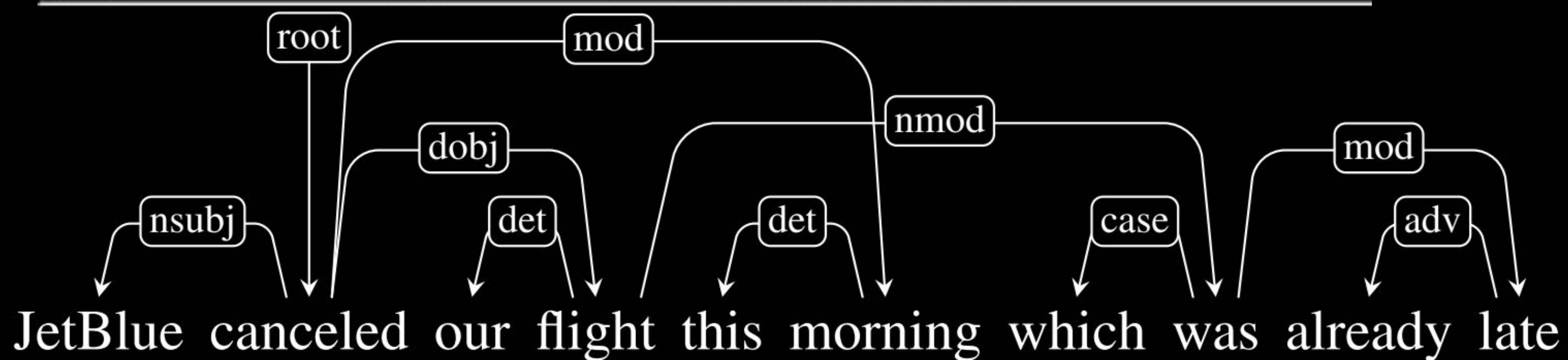
Complex and slow but leads to state of the art. Now done with neural models.



(From SLP 3rd ed., Jurafsky and Martin 2018)

Relation to Semantic Roles

Thematic Role	Definition
AGENT	The volitional causer of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional causer of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
CONTENT	The proposition or content of a propositional event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event



(13.3)

(From SLP 3rd ed., Jurafsky and Martin 2018)

Semantics Avalanche

Key Takeaways:

- Words have many meanings.
  - Context is key
  - Selectors can represent context
- Verbs can been seen as functions (predicates) that take arguments.
  - Arguments fulfill semantic roles
- Words have implicit relationships with each other in given sentences.
  - Dependency Parsing: each word has one head
  - Easily constructed through 3 actions of shift-reduce parsing.
- There is an interplay between word meaning and sentence structure