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

CSE392 - Spring 2019
Special Topic in CS
Tasks

- Word Sense Disambiguation
- Dependency Parsing
- Semantic Role Labeling

- 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

Traditionally:
- Probabilistic models
- Discriminant Learning: e.g. Logistic Regression
- Transition-Based Parsing
- Graph-Based Parsing

Current:
Recurrent Neural Network
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

<table>
<thead>
<tr>
<th>Wordform</th>
<th>Lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>banks</td>
<td>bank</td>
</tr>
<tr>
<td>sung</td>
<td>sing</td>
</tr>
<tr>
<td>duermes</td>
<td>dormir</td>
</tr>
</tbody>
</table>
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...

  Sense 2:  • "...as agriculture burgeons on the east bank 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)

**Homonymy**

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

- \( \text{bank}_1 \): financial institution, \( \text{bank}_2 \): sloping land
- \( \text{bat}_1 \): club for hitting a ball, \( \text{bat}_2 \): nocturnal flying mammal

1. Homographs (\( \text{bank}/\text{bank}, \text{bat}/\text{bat} \))
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)
He put the **port** on the ship.

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

He walked along the **port** next to the steamer.
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)
- **n.2** port wine (sweet dark-red dessert wine originally from Portugal)
- **n.3**, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)
- larboard, **n.4** (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)
- interface, **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))
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)
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))
An Approach to WSD

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

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

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>w/ sels</th>
<th>mfs</th>
<th>tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>87.9</td>
<td>91.7</td>
<td>80.9</td>
<td>2559</td>
</tr>
<tr>
<td>verb</td>
<td>83.3</td>
<td>83.7</td>
<td>76.5</td>
<td>2292</td>
</tr>
<tr>
<td>both</td>
<td>85.7</td>
<td>87.9</td>
<td>78.8</td>
<td>4851</td>
</tr>
</tbody>
</table>

Accuracy over SemEval-2007: Task 17.
## Supervised Selectors

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>w/ sels</th>
<th>mfs</th>
<th>tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>87.9</td>
<td>91.7</td>
<td>80.9</td>
<td>2559</td>
</tr>
<tr>
<td>verb</td>
<td>83.3</td>
<td>83.7</td>
<td>76.5</td>
<td>2292</td>
</tr>
<tr>
<td>both</td>
<td>85.7</td>
<td>87.9</td>
<td>78.8</td>
<td>4851</td>
</tr>
</tbody>
</table>

**Accuracy over SemEval-2007: Task 17.**

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>w/ sels</th>
<th>mfs</th>
<th>tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>68.5</td>
<td>72.1</td>
<td>54.1</td>
<td>1766</td>
</tr>
<tr>
<td>verb</td>
<td>72.0</td>
<td>72.4</td>
<td>57.9</td>
<td>1927</td>
</tr>
<tr>
<td>adjective</td>
<td>49.4</td>
<td>53.4</td>
<td>54.7</td>
<td>148</td>
</tr>
<tr>
<td>all</td>
<td>69.4</td>
<td>71.5</td>
<td>56.1</td>
<td>3841</td>
</tr>
</tbody>
</table>

**Accuracy over seneval-3 Lexical Sample.**
(fine-grained senses compared to SemEval)
Why Are Selectors Effective?

Sets of selectors tend to vary extensively by word sense:

<table>
<thead>
<tr>
<th>bill-n.1</th>
<th>bill-n.2</th>
<th>bill-n.3</th>
<th>occur-v.1</th>
<th>occur-v.2</th>
<th>occur-v.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>bill</td>
<td>bill</td>
<td>market</td>
<td>be</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>it</td>
<td>staff</td>
<td>system</td>
<td>happen</td>
<td>get</td>
<td>look</td>
</tr>
<tr>
<td>legislation</td>
<td>system</td>
<td>paper</td>
<td>occur</td>
<td>Come</td>
<td>break</td>
</tr>
<tr>
<td>system</td>
<td>money</td>
<td>note</td>
<td>go</td>
<td>have</td>
<td>remove</td>
</tr>
<tr>
<td>program</td>
<td>time</td>
<td>bill</td>
<td>take</td>
<td>try</td>
<td>find</td>
</tr>
<tr>
<td>law</td>
<td>it</td>
<td>bond</td>
<td>work</td>
<td>lead</td>
<td>get</td>
</tr>
<tr>
<td>plan</td>
<td>tax</td>
<td>stock</td>
<td>come</td>
<td>listen</td>
<td>place</td>
</tr>
<tr>
<td>you</td>
<td>work</td>
<td>debt</td>
<td>see</td>
<td>work</td>
<td>keep</td>
</tr>
<tr>
<td>measure</td>
<td>rent</td>
<td>rate</td>
<td>have</td>
<td>be</td>
<td>stick</td>
</tr>
<tr>
<td>project</td>
<td>tuition</td>
<td>report</td>
<td>change</td>
<td>belong</td>
<td>stop</td>
</tr>
</tbody>
</table>
• 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”)

<table>
<thead>
<tr>
<th>bill-n.1</th>
<th>bill-n.2</th>
<th>bill-n.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>bill</td>
<td>bill</td>
<td>market</td>
</tr>
<tr>
<td>it</td>
<td>staff</td>
<td>system</td>
</tr>
<tr>
<td>legislation</td>
<td>system</td>
<td>paper</td>
</tr>
<tr>
<td>system</td>
<td>money</td>
<td>note</td>
</tr>
<tr>
<td>program</td>
<td>time</td>
<td>bill</td>
</tr>
<tr>
<td>law</td>
<td>it</td>
<td>bond</td>
</tr>
<tr>
<td>plan</td>
<td>tax</td>
<td>stock</td>
</tr>
<tr>
<td>you</td>
<td>work</td>
<td>debt</td>
</tr>
<tr>
<td>measure</td>
<td>rent</td>
<td>rate</td>
</tr>
<tr>
<td>project</td>
<td>tuition</td>
<td>report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>occur-v.1</th>
<th>occur-v.2</th>
<th>occur-v.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>be</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>happen</td>
<td>get</td>
<td>look</td>
</tr>
<tr>
<td>occur</td>
<td>Come</td>
<td>break</td>
</tr>
<tr>
<td>go</td>
<td>have</td>
<td>remove</td>
</tr>
<tr>
<td>take</td>
<td>try</td>
<td>find</td>
</tr>
<tr>
<td>work</td>
<td>lead</td>
<td>find</td>
</tr>
<tr>
<td>come</td>
<td>listen</td>
<td>place</td>
</tr>
<tr>
<td>see</td>
<td>work</td>
<td>keep</td>
</tr>
<tr>
<td>have</td>
<td>be</td>
<td>stick</td>
</tr>
<tr>
<td>change</td>
<td>belong</td>
<td>stop</td>
</tr>
</tbody>
</table>
Tasks

- Word Sense Disambiguation
- Dependency Parsing
- Semantic Role Labeling

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

- 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

I prefer the morning flight through Denver

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

(13.1) I prefer the morning flight through Denver

(prefer
  / 
 I flight
  / 
the morning Denver

(root
  / 
  dobj
  / 
  det
  / 
nmod

  nsubj
  / 
  I

  nmod
  / 
  through

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

## Clausal Argument Relations

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUBJ</td>
<td>Nominal subject</td>
</tr>
<tr>
<td>DOBJ</td>
<td>Direct object</td>
</tr>
<tr>
<td>IOBJ</td>
<td>Indirect object</td>
</tr>
<tr>
<td>CCOMP</td>
<td>Clausal complement</td>
</tr>
<tr>
<td>XCOMP</td>
<td>Open clausal complement</td>
</tr>
</tbody>
</table>

## Nominal Modifier Relations

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMOD</td>
<td>Nominal modifier</td>
</tr>
<tr>
<td>AMOD</td>
<td>Adjectival modifier</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Numeric modifier</td>
</tr>
<tr>
<td>APPOS</td>
<td>Appositional modifier</td>
</tr>
<tr>
<td>DET</td>
<td>Determiner</td>
</tr>
<tr>
<td>CASE</td>
<td>Prepositions, postpositions and other case markers</td>
</tr>
</tbody>
</table>

## Other Notable Relations

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONJ</td>
<td>Conjunct</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
</tr>
</tbody>
</table>

*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

<table>
<thead>
<tr>
<th>Relation</th>
<th>Examples with <em>head</em> and <em>dependent</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUBJ</td>
<td>United <em>canceled</em> the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>United <em>diverted</em> the <em>flight</em> to Reno.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>We <em>booked</em> her the first <em>flight</em> to Miami.</td>
</tr>
<tr>
<td>NMOD</td>
<td>We <em>booked</em> her the <em>flight</em> to Miami.</td>
</tr>
<tr>
<td>AMOD</td>
<td>We took the <em>morning flight</em>.</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Book the <em>cheapest flight</em>.</td>
</tr>
<tr>
<td>APPOS</td>
<td>Before the storm JetBlue canceled <em>1000 flights</em>.</td>
</tr>
<tr>
<td>DET</td>
<td><em>United</em>, a <em>unit</em> of UAL, matched the fares.</td>
</tr>
<tr>
<td>DET</td>
<td>The <em>flight</em> was canceled.</td>
</tr>
<tr>
<td>CONJ</td>
<td>We <em>flew</em> to Denver and <em>drove</em> to Steamboat.</td>
</tr>
<tr>
<td>CC</td>
<td>We <em>flew</em> to Denver <em>and</em> <em>drove</em> to Steamboat.</td>
</tr>
<tr>
<td>CASE</td>
<td>Book the flight <em>through</em> Houston.</td>
</tr>
</tbody>
</table>

*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.

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

**Figure 13.3**  Examples of core Universal Dependency relations.

*(From SLP 3rd ed., Jurafsky and Martin 2018)*
Dependency Parsing -- Verbal Predicates

United canceled the morning flights to Houston

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- Verbal Predicates

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

(13.2)

(From SLP 3rd ed., Jurafsky and Martin 2018)
to_call_off("United", "the morning flights to Houston")

\[(13.2)\]

(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: $G = [(V1, A1), (V1, A2), \ldots]$ (vertices and arcs)

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

(13.2) United canceled the morning flights to Houston

(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 \((w_1, w_2, \ldots)\) of sentence
- **A**: set of dependency arcs, initialized empty
- **T**: Actions, given \(w_i\) (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 $\leftarrow \{\text{[root], [words], []}\}$ ; initial configuration

**while** state not final

$\text{t} \leftarrow \text{ORACLE(state)}$ ; choose a transition operator to apply

state $\leftarrow \text{APPLY}(t, \text{state})$ ; 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.

<table>
<thead>
<tr>
<th>Stack</th>
<th>Word List</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[root, book, me]</td>
<td>[the, morning, flight]</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Stack</th>
<th>Word List</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[root, book]</td>
<td>[the, morning, flight]</td>
<td>(book → me)</td>
</tr>
</tbody>
</table>

(From SLP 3rd ed., Jurafsky and Martin 2018)
## Transition-based Dependency Parsing

<table>
<thead>
<tr>
<th>Step</th>
<th>Stack</th>
<th>Word List</th>
<th>Action</th>
<th>Relation Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[root]</td>
<td>[book, me, the, morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[root, book]</td>
<td>[me, the, morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[root, book, me]</td>
<td>[the, morning, flight]</td>
<td>RIGHTARC</td>
<td>(book → me)</td>
</tr>
<tr>
<td>3</td>
<td>[root, book]</td>
<td>[the, morning, flight]</td>
<td>SHIF</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[root, book, the]</td>
<td>[morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>[root, book, the, morning]</td>
<td>[flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>[root, book, the, morning, flight]</td>
<td>[]</td>
<td>LEFTARC</td>
<td>(morning ← flight)</td>
</tr>
<tr>
<td>7</td>
<td>[root, book, the, flight]</td>
<td>[]</td>
<td>LEFTARC</td>
<td>(the ← flight)</td>
</tr>
<tr>
<td>8</td>
<td>[root, book, flight]</td>
<td>[]</td>
<td>RIGHTARC</td>
<td>(book → flight)</td>
</tr>
<tr>
<td>10</td>
<td>[root]</td>
<td>[]</td>
<td>Done</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 13.7** Trace of a transition-based parse.

*(From SLP 3rd ed., Jurafsky and Martin 2018)*
Dependency Parsing -- How to Represent?

A Graph: \( G = [(V_1, A_1), (V_1, A_2), \ldots] \) (vertices and arcs)

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

(13.2) United canceled the morning flights to Houston

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- How to Represent?

A Graph: $G = [(V_1, A_1), (V_1, A_2), \ldots]$ (vertices and arcs)

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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) United canceled the morning flights to Houston

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- How to Represent?

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

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- How to Represent?

A Graph: \( G = [(V_1, A_1), (V_1, A_2), \ldots] \) (vertices and arcs)

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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

JetBlue canceled our flight this morning which was already late

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- How to Represent?

A Graph: \( G = [(V_1, A_1), (V_1, A_2), \ldots] \) (vertices and arcs)
Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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) JetBlue canceled our flight this morning which was already late

(From SLP 3rd ed., Jurafsky and Martin 2018)
Dependency Parsing -- How to Represent?

A Graph: \( G = [(V_1, A_1), (V_1, A_2), \ldots] \) (vertices and arcs)

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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.

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.
Graph-based Approaches

A Graph: \( G = [(V_1, A_1), (V_1, A_2), \ldots] \) (vertices and arcs)

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

General Idea: Search through all possible trees and pick best.
Graph-based Approaches

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

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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: \( G = [(V1, A1), (V1, A2), \ldots] \) (vertices and arcs)

Restrictions:
1) Single designated ROOT with no incoming arcs
2) Every vertex only has one head (parent, governor); 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

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

JetBlue canceled our flight this morning which was already late

(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