Outline

- Lexicalized and state-split constituency parsing (slides from last time)
- Dependency representation
- Contrast with constituency
- Projectivity

Lexicalized Parsing

```
S(ran)
  |__________________________|
  |                          |
NP(dog)                        VP(ran)
  |__________________________|
  |                          |
DT(the) the NN(dog) dog       TO(to)
  |__________________________|
  |                          |
VBD(ran) ran                  DT(the) the
  |__________________________|
  |                          |
NP(house)                      NN(house)
the house to ran the dog
```

Administrivia

- Project 1 due Thursday at 9:30am
Dependency Parsing

- Dependency syntax: syntactic structure is defined by dependencies
  - Head (parent, governor) connected to dependent (child, modifier)
  - Each word has exactly one parent except for the ROOT symbol
  - Dependencies must form a directed acyclic graph

```
ROOT
DT the
NN dog
VBD ran
TO to
DT the
NN house
```

- Still a notion of hierarchy!

```
VBD ran
DT the
NN dog
TO to
DT the
NN house
```

- Can still derive constituents (subtrees)

```
```

Dependency Parsing

- Can label dependencies according to syntactic function
- Major source of ambiguity is in the structure, so we focus on that more (labeling separately with a classifier works pretty well)

```
det
ds
prep
pobj

DT the
NN dog
VBD ran
TO to
DT the
NN house
```

```
```

Dependency vs. Constituency: PP Attachment

- Constituency: several rule productions need to change
Dependency vs. Constituency: PP Attachment

- Dependency: one word (with) assigned a different parent

- More predicate-argument focused view of syntax

- “What’s the main verb of the sentence? What is its subject and object?” — easier to answer under dependency parsing

---

Dependency vs. Constituency: Coordination

- Constituency: ternary rule NP -> NP CC NP

- Dependency: first item is the head

- Coordination is decomposed across a few arcs as opposed to being a single rule production as in constituency

- Can also choose and to be the head

- Both cases: headword doesn’t really represent the phrase

---

Stanford Dependencies

- Designed to be practically useful for relation extraction

Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas

- Standard

- Collapsed
Dependency vs. Constituency

- Dependency is often more useful in practice (models predicate argument structure)
- Slightly different representational choices:
  - PP attachment is better modeled under dependency
  - Coordination is better modeled under constituency
- Dependency parsers are easier to build: no “grammar engineering”, no unaries, easier to get structured discriminative models working well
- Dependency parsers are usually faster
- Dependencies are more universal cross-lingually

Universal Dependencies

- Annotate dependencies with the same representation in many languages

Projectivity

- What conditions have to hold for things to be tree-shaped?

  - Any subtree is a contiguous span of the sentence <-> tree is projective

Projectivity

- Projective <-> no “crossing” arcs
- Crossing arcs:
- Extrapolation: A hearing on the issue is scheduled today. is projective

credit: Language Log
More extraposition

Time expressions can go a lot of places in sentences!

Many trees in other languages are nonprojective

Some other formalisms (that are harder to parse in), most useful one is 1-Endpoint-Crossing

Number of trees produceable under different formalisms:

<table>
<thead>
<tr>
<th></th>
<th>Arabic</th>
<th>Czech</th>
<th>Danish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projective</td>
<td>1297 (88.8%)</td>
<td>55872 (76.8%)</td>
<td>4379 (84.4%)</td>
</tr>
<tr>
<td>Sentences</td>
<td>1460</td>
<td>72703</td>
<td>5190</td>
</tr>
</tbody>
</table>

1-Endpoint-Crossing: for any edge, all edges that cross it share an endpoint

Captures most cases, still efficient parsing algorithms

John was not as good for the job as Kate

True

False: hearing -> on