Introduction to NLP
Introduction to Dependency Trees

Prof. Reut Tsarfaty
Bar Ilan University
Introduction to Dependency Parsing

- The purpose of Syntactic Structures:
  - Encode *Predicate Argument* Structures
  - *Who Does What to Whom?* (When, Where, Why...)
Introduction to Dependency Parsing

- The purpose of Syntactic Structures:
  - Encode *Predicate Argument* Structures
  - *Who Does What to Whom?* (When, Where, Why...)

- Properties of Dependency Structures:
  - Defined as (labeled) binary relations between words
  - Reflect a long linguistic (European) tradition
  - Explicitly represent *Argument Structure*
Where Do Dependency Trees Come From?

```
S
  NP  VP
  workers
  VP
    V  NP
    dumped  sacks
    PP
      P  NP
      into  bins
```
Where Do Dependency Trees Come From?

S/dumped

NP/workers
  workers

VP/dumped

NP/sacks
  sacks

VP/dumped

V/dumped

P/into

PP/into

NP/bins

into

bins
Where do Dependency Trees Come From?

dumped

workers

workers

dumped

dumped

dumped

sacks

into

into

bins

bins
Where do Dependency Trees Come From?

dumped

workers  dumped

workers

dumped  into

sacks  into

into

sacks  bins

into  bins
Where do Dependency Trees Come From?

ROOT

dumped

workers sacks into bins
Unlabeled Dependency Tree:

Labeled Dependency Tree:
Representation: Functional vs. Lexical

Functional Dependencies:

-ROOT-

\( \text{dumped} \)

subj \( \text{workers} \)

dobj \( \text{sacks} \)

prep \( \text{into} \)

pobj \( \text{bins} \)

Lexical Dependencies:

-ROOT-

\( \text{dumped} \)

subj \( \text{workers} \)

dobj \( \text{sacks} \)

nmod \( \text{bins} \)

pobj \( \text{case} \)

into
Discussions: Options and Schemes

**Vertical vs. Horizontal Representation**
http://nlp.stanford.edu:8080/corenlp/

**The Universal Dependencies Initiative**
https://universaldependencies.org/
A labeled dependency tree is a labeled directed tree $T$:
- a set $V$ of nodes, labeled with words (including ROOT)
- a set $A$ of arcs, labeled with dependency types
- a linear precedence order $<$ on $V$

Notation:
- Arc $\langle v_1, v_2 \rangle$ connects head $v_1$ with dep $v_2$
- Arc $\langle v_1, l, v_2 \rangle$ connects head $v_1$ with dep $v_2$ with label $l \in L$
- A node $v_0$ (ROOT) serves as a unique root of the tree
Properties of Dependency Trees

A dependency $T$ tree is:

- **connected:**
  For every node $i$ there is a node $j$ such that $i \rightarrow j$ or $j \rightarrow i$

- **acyclic:**
  If $i \rightarrow j$ then not $j \rightarrow \ast i$

- **single head:**
  If $i \rightarrow j$ then not $k \rightarrow j$ for any $k \neq i$

- **projective:**
  If $i \rightarrow j$ then $i \rightarrow \ast k$ for any $k$ such that $i < k < j$
Non-Projective Dependency Trees

Figure 1: A projective dependency graph.

Figure 2: Non-projective dependency graph.
Non-Projective Dependency Trees

Many parsing algorithms are restricted to projective dependency trees.

Is this a problem?

Statistics from CoNLL-X Shared Task 2006

- NPD = Non-projective dependencies
- NPS = Non-projective sentences

<table>
<thead>
<tr>
<th>Language</th>
<th>%NPD</th>
<th>% NPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch</td>
<td>5.4</td>
<td>36.4</td>
</tr>
<tr>
<td>German</td>
<td>2.3</td>
<td>27.8</td>
</tr>
<tr>
<td>Czech</td>
<td>1.9</td>
<td>23.2</td>
</tr>
<tr>
<td>Slovene</td>
<td>1.9</td>
<td>22.2</td>
</tr>
<tr>
<td>Portuguese</td>
<td>1.3</td>
<td>18.9</td>
</tr>
<tr>
<td>Danish</td>
<td>1.0</td>
<td>15.6</td>
</tr>
</tbody>
</table>

We will (mostly) focus on projective dependencies.
Let's Analyse!

The cat sat on the mat.
Let's Analyse!

The cat is on the mat.
Let’s Analyse!

The cat is currently sitting on the mat.
Let’s Analyse!

The cat, which I met, is sitting on the mat.
Let’s Analyse!

The dog and the cat sat on the big and fluffy mat.
Let’s Analyse!

The dog and the cat sat on the big and fluffy mat

You should know how to read/analyse these!
Our Plan:

Today we saw:
- Two kinds of formal syntactic representations
- Formally clean, complementary traits
- Theoretically (somewhat) compromised

Next class:
- Models and Algorithms for
  - Phrase-Structure Parsing
  - Dependency Parsing
- Evaluation Metrics