Introduction to NLP

Dependency Parsing

Prof. Reut Tsarfaty
Bar Ilan University

November 16, 2020
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?

Remember Phrase ‘Heads’?

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

Step 1: Percolate ‘Heads’

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

Step 1: Percolate ‘Heads’

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

Step 1: Percolate ‘Heads’

S/dumped

NP/workers
  workers

VP/dumped

V/dumped
  dumped

NP/sacks
  sacks

PP/into
  P/into
    into
  NP/bins
    bins
Where do Dependency Trees Come From?

Step 2: Remove Phrase Labels

```
dumped
  /   \\
workers     dumped
         /   \\
workers     dumped
        /   |
       dumped sacks
      /   |
     dumped sacks
    /     |
   dumped into bins
  /     |
 into bins
```
Where do Dependency Trees Come From?

Step 3: Remove Duplicate Nodes (retain highest)
Where do Dependency Trees Come From?

We are left with a tree of **Bi-Lexical** Dependencies

```
          ─ROOT─
             |
        dumped
     ┌───┐  ┌───┐  ┌───┐
workers sacks into bins
```
Representation: Labeled vs. Unlabeled

Unlabeled Dependency Tree:

-LIGHTTEXT-ROOT-

dumped

workers \hspace{0.1cm} sacks \hspace{0.1cm} into

bins

Labeled Dependency Tree:

-LIGHTTEXT-ROOT-

root dumped

subj workers
dobj sacks
prep into
pobj bins
Representation: Functional vs. Lexical

Functional Dependencies:

-ROOT-
  \-root-
  \-dumped-
  subj workers
dobj sacks
prep into
pobj bins

Lexical Dependencies:

-ROOT-
  \-root-
  \-dumped-
  subj workers
dobj sacks
nmod bins
  case
into
Representation: Options and Schemes

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

The Universal Dependencies Initiative
https://universaldependencies.org/
Let’s Analyse!

John loves Mary.
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!
Let’s Analyse!

He said that the boy who was wearing the blue shirt with the white pockets has left the building.
Let’s Analyse!

a large pile of carrots and peas was closely guarded by dogs.
Tricky Cases!

They wanted to buy cakes and eat them on the road.
Tricky Cases!

I bought soda and pizza for John and Mary.
Tricky Cases!

I bought soda and pizza for 4 and 57 cents.
Tricky Cases!

I ordered five books but received four.
Tricky Cases!

While Sue has many toys, Alice doesn’t have any.
Tricky Cases!

Cut, chop and peel the tomatoes
Tricky Cases!

Cut the tomatoes. Put in a bowl.
We introduced two different syntactic representations:

**Phrase-Structure Trees**
- Hierarchical
- Mark labeled spans (S,NP,VP,..)
- Assume Context-Free Grammars

**Dependency Trees**
- Hierarchical
- Mark labeled dependencies ((wi,\textit{subj},wj),(wi,\textit{obj},wk)... )
- Assume Non-Crossing Dependencies

**Our task:** Parse NL sentences into trees
Introducing the Parsing Problem

The Parsing Task

Sentence $x \rightarrow \text{Parser} \rightarrow \text{Parse-Tree } t$

The Objective Function

$$t^* = \arg\max_{t \in \text{GEN}(x)} \text{Score}(t)$$

Modeling Decisions

- $\text{GEN}(x)$: How to construct all candidate trees for $x$?
- $\text{Score}(t)$: How to score any candidate tree for $x$?
- $\arg\max_{t \in \text{GEN}(x)}$: How to find the best tree for $x$?
How To Solve This

We want to solve the parsing task...

- ... for Phrase-Structure Trees
- ... for Dependency-Trees

To Be Continued...

Next Week
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 $j \rightarrow^* i$

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

- **projective:**
  If $i \rightarrow j$ then $i \rightarrow^* k$ 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.