Introduction to NLP
Data-Driven Dependency Parsing

Prof. Reut Tsarfaty
Bar Ilan University
Dependency Trees: Introduction

- 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*
Dependency Trees: Formal Definition

- 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

![Dependency Tree Diagram]
Properties of Dependency Trees

A dependency $T$ tree is:

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

- **acyclic:**
  If $i \to j$ then not $j \to^* i$

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

- **projective:**
  If $i \to j$ then $i \to^* k$ 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.
Draw the Lexical and Functional Dependency Trees for:

- The cat sat on the mat.
- The cat is on the mat.
- The cat is currently sitting on the mat.
- The cat, which I met, is sitting on the mat.
- The dog and the cat sat on the big and fluffy mat.
Do Try This At Home!

Draw the Lexical and Functional Dependency Trees for:

- The cat sat on the mat.
- The cat is on the mat.
- The cat is currently sitting on the mat.
- The cat, which I met, is sitting on the mat.
- The dog and the cat sat on the big and fluffy mat.

Sample Answers are on the website!
Our Plan:

**Last time:**
- Two kinds of formal syntactic representations
- Formally clean, complementary traits
- Theoretically (somewhat) compromised

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

The Parsing Objective:

\[ y^* = \arg\max_{y \in \text{GEN}(x)} \text{Score}(y) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>?</td>
</tr>
<tr>
<td>Train</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Generative vs. Discriminative Modeling

From a Probabilistic to a Discriminative Model:

\[ t^* = \arg\max_{t \in \{t \mid Y(t) = x\}} P(t) \]

\[ = \arg\max_{t \in \{t \mid Y(t) = x\}} \prod_{r \in t} P(r)^{\text{count}(r \in t)} \]

\[ = \arg\max_{t \in \{t \mid Y(t) = x\}} -\log(\prod_{r \in t} P(r)^{\text{count}(r \in t)}) \]

\[ = \arg\max_{t \in \{t \mid Y(t) = x\}} \sum_{r \in t} -\log P(r) \times \text{count}(r \in t) \]

\[ = \arg\max_{t \in \{t \mid Y(t) = x\}} \sum_{f} \theta_f \times \text{count}(f) \]

\[ = \arg\max_{t \in \{t \mid Y(t) = x\}} w^T \Phi(t) \]
Today: Introduction to Dependency Parsing

The Objective Function:

\[ y^* = \arg\max_{y \in \text{GEN}(x)} w^T \Phi(x, y) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>?</td>
</tr>
<tr>
<td>Train</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

\[ y^* = \text{argmax}_{y \in \text{GEN}(x)} \mathbf{w}^T \Phi(x, y) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>?</td>
</tr>
<tr>
<td>Train</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

\[ y^* = \underset{y \in \text{GEN}(x)}{\text{argmax}} \ w^T \phi(x, y) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>?</td>
</tr>
<tr>
<td>Train</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

\[ y^* = \arg\max_{y \in GEN(x)} \{ w^T \phi(y) \} \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>?</td>
</tr>
<tr>
<td><strong>Train</strong></td>
<td>?</td>
</tr>
<tr>
<td><strong>Evaluation</strong></td>
<td><strong>Attachment Scores</strong></td>
</tr>
</tbody>
</table>
Our Modeling Tasks:

\[ t = \arg\max_{t \in GEN(x)} w^T \Phi(t) \]

- **GEN**: How do we generate all \( t \)?
- **argmax**: How do we search through all \( t \)?
- **\( \Phi \)**: How do we featurize \( \Phi(t) \)?
- **\( w \)**: How do we learn the weights \( w \)?
A Note on Evaluation Metrics

- **Unlabeled Attachment Scores (UAS)**
  The percentage of identical arcs from the total number or arcs in the tree
  \[ UAS = \frac{A_{\text{intersect}(i,j)}}{n} \]

- **Labeled Attachment Scores (LAS)**
  The percentage of identical arcs with identical labels from the total number or arcs in the tree
  \[ LAS = \frac{A_{\text{intersect}(i,l,j)}}{n} \]

- **Root Accuracy**
  The percentage of sentences with correct root dependency

- **Exact Match**
  The percentage of sentences with parses identical to gold
Modeling Methods

- **Conversion-Based**
  - Convert Phrase-Structure to Dependency Trees

- **Grammar-Based**
  - Generative methods based on PCFGs

- **Graph-Based**
  - Globally Optimised, Restricted features

- **Transition-Based**
  - Locally Optimal, Unrestricted features

- **Neural-Based**
Modeling Methods (1)

- **Conversion-Based**: Convert PS trees Using a Head Table
- Grammar-Based
- Graph-Based
- Transition-Based
**Conversion-Based**: Convert PS trees Using a Head Table

| VP   | → | VBD VBN MD VBZ VB VBG VBP VP |
| NP   | ← | NN NX JJR CD JJ JJS RB       |
| ADJP | ← | NNS QP NN ADVP JJ VBN VBG    |
| ADVP | → | RB RBR RBS FW ADVP TO CD JJR |
| S    | ← | VP S SBAR ADJP UCP NP       |
| SQ   | ← | VBZ VBD VBP VB MD PRD VP SQ  |
| SBAR | ← | S SQ SINV SBAR FRAG IN DT   |
Modeling Methods (2)

- Conversion-Based
  - Grammar-Based
  - Graph-Based
  - Transition-Based
Grammar-Based Dependency Parsing

The Basic Idea

- Treat bi-lexical dependencies as constituents
- Decode using chart based algorithm (e.g., CKY)
- Learn using standard MLE methods
- Evaluate over the set of resulting dependencies as usual
Grammar-Based Dependency Parsing

The Basic Idea
- Treat bi-lexical dependencies as constituents
- Decode using chart based algorithm (e.g., CKY)
- Learn using standard MLE methods
- Evaluate over the set of resulting dependencies as usual

Relevant Studies
- Original version: [Hays 1964]
- Link Grammar: [Sleator and Temperley 1991]
- Earley-style left-corner: [Lombardo and Lesmo 1996]

http://cs.jhu.edu/~jason/papers/eisner.coling96.pdf
The Objective Function:

\[ t^* = \arg\max_{t \in \text{GEN}(x)} P(t) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>PCFG</td>
</tr>
<tr>
<td>Decoder</td>
<td>Adapted CKY</td>
</tr>
<tr>
<td>Trainer</td>
<td>Smoothed MLE</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Modeling Methods (3)

✓ Conversion-Based
✓ Grammar-Based
▶ Graph-Based
▶ Transition-Based
Graph-Based Dependency Parsing

The Basic Idea

- Define a global Arc-Factored model
- Treat the search as an MST problem
- Treat the learning as a classification problem
- Evaluate over the set of gold dependencies as usual
Graph-Based Dependency Parsing

The Basic Idea

- Define a global Arc-Factored model
- Treat the search as an MST problem
- Treat the learning as a classification problem
- Evaluate over the set of gold dependencies as usual
Step 1: Defining the Arc Factored Model

\[ t^* = \arg \max_{t \in \text{GEN}(V)} \Phi(t) = \arg \max_{t \in \text{GEN}(V)} \sum_{(i \rightarrow j) \in t} \phi_{\text{arc}}(i \rightarrow j) \]
Graph-Based Dependency Parsing

Step 2: Defining Feature Templates

- **ϕ_i** (had, OBJ, effect)
- **w_i**
- **unihead**
- **unidep**
- **uniheadpos**
- **unideppos**
- **bigram**
- **bigrampos**
- **bigramlabel**
- **bigramposlabel**
- **inbetween**
Graph-Based Dependency Parsing

Step 2: Defining Feature Templates

<table>
<thead>
<tr>
<th>Name</th>
<th>$\phi_i(\text{had,OBJ, effect})$</th>
<th>$w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram head</td>
<td>“had”</td>
<td>$w_{\text{unihead}}$</td>
</tr>
<tr>
<td>Unigram dep</td>
<td>“effect”</td>
<td>$w_{\text{unidep}}$</td>
</tr>
<tr>
<td>Unigram head pos</td>
<td>VB</td>
<td>$w_{\text{uniheadpos}}$</td>
</tr>
<tr>
<td>Unigram dep pos</td>
<td>NN</td>
<td>$w_{\text{unidepos}}$</td>
</tr>
<tr>
<td>Bigram head-dep</td>
<td>“had-effect”</td>
<td>$w_{\text{bigram}}$</td>
</tr>
<tr>
<td>Bigram headpos-depos</td>
<td>VB-NN</td>
<td>$w_{\text{bigrampos}}$</td>
</tr>
<tr>
<td>Labeled Bigram head-dep</td>
<td>“had-OBJ-effect”</td>
<td>$w_{\text{bigramlabel}}$</td>
</tr>
<tr>
<td>Labeled Bigram headpos-depos</td>
<td>VB-obj-NN</td>
<td>$w_{\text{bigramposlabel}}$</td>
</tr>
<tr>
<td>In-Between pos</td>
<td>VB-IN-NN</td>
<td>$w_{\text{inbetween}}$</td>
</tr>
</tbody>
</table>
Step 3: Online Learning

Perceptron

1. \( w \leftarrow 0 \)
2. for \( t = 1 \ldots T, i = 1 \ldots N \) do
3. \( z_i = \arg \max_{y \in \mathcal{Y}} g(y; x_i, w) \)
4. \( \text{gold} \leftarrow \sum \{a | y_i(a) = 1\} \phi(x_i, a) \)
5. \( \text{best} \leftarrow \sum \{a | z_i(a) = 1\} \phi(x_i, a) \)
6. \( w \leftarrow w + \text{gold} - \text{best} \)
7. return \( w \)
Step 3: Online Learning

Perceptron

▶ Theory:
- If possible, will learn to separate the correct structure from the incorrect structures
- I.e. find \( w \) that assigns higher scores to \( y_i \) then any \( y \in \mathcal{Y} \)

▶ Practice:
- Training requires many inferences
- Computing feature values is time consuming
- Averaged Perceptron variant preferred
Graph-Based Dependency Parsing

Step 4: Finding the Max-Spanning Tree

The Chu-Liu-Edmonds Algorithm

Runtime complexity: $O(n^2)$
Graph-Based Dependency Parsing

Step 3: Online Learning
Perceptron/MIRA (Margin Infused Relaxed Algorithm)

Step 4: Max-Spanning Tree Decoding
The Chu-Liu-Edmonds Algorithm (CLE)

http://repository.upenn.edu/cgi/viewcontent.cgi?action=1056&context=cis_reports
Graph-Based Dependency Parsing

The Objective Function:

\[ t^* = \arg\max_{\{t|\in GEN(x)\}} \sum_{a\in arcs(t)} w^T \Phi(a) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Graph-Based</td>
</tr>
<tr>
<td></td>
<td>Arc-Factored</td>
</tr>
<tr>
<td>Decoder</td>
<td>MST/CLE (O(n^2))</td>
</tr>
<tr>
<td>Trainer</td>
<td>Perceptron/MIRA</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Modeling Methods (4)

✓ Conversion-Based
✓ Grammar-Based
✓ Graph-Based
► Transition-Based
Transition-Based Dependency Parsing

The Basic Idea

- Define a transition system
- Define an Oracle Algorithm for Decoding
- Approximate the Oracle Algorithm via Learning
- Evaluate over Dependency Arcs as Usual

http://stp.lingfil.uu.se/~nivre/docs/BeyondMaltParser.pdf
Transition-Based Dependency Parsing

Defining Configurations
A parser **Configuration** is a triplet \( c = (S, Q, A) \), where

- \( S = \) a stack \([..., w_i]_S\) of partially processed nodes
- \( Q = \) a queue \([w_j, ...]_Q\) of remaining input nodes
- \( A = \) a set of labeled arcs \((w_i, l, w_j)\)

**Initialization:**
- \( c_0 = ([w_0]_S, [w_1, ..., w_n]_Q, \{\}) \)
  
  Note: \( w_0 = \text{ROOT} \)

**Termination:**
- \( c_t = ([w_0]_S, [], _Q, A) \)
Transition-Based Dependency Parsing

Defining Transitions

▶ **Shift:**
\[
([...]_s, [w_i, ...]_q, A) \rightarrow ([..., w_i]_s, [...]_q, A)
\]

▶ **Arc-Left(\(l\)):**
\[
([..., w_i, w_j]_s, Q, A) \rightarrow ([..., w_j]_s, Q, A \cup (w_j, l, w_i))
\]

▶ **Arc-Right(\(l\)):**
\[
([..., w_i, w_j]_s, Q, A) \rightarrow ([..., w_i]_s, Q, A \cup (w_i, l, w_j))
\]
Transition-Based Dependency Parsing

Demo Deck
Transition-Based Dependency Parsing

Deterministic Parsing

Given an oracle $O$ that correctly predicts the next transition $O(c)$, parsing is deterministic:

$$\text{PARSE}(w_1, ..., w_n)$$

1. $c \leftarrow ([w_0]_S, [w_1, ..., w_n]_Q)$
2. while $Q_c \neq []$ or $|S_c| = 1$
3. $t \leftarrow O(c)$
4. $c \leftarrow t(c)$
5. return $T = (w_0, w_1, ..., w_n, A_c)$
Transition-Based Dependency Parsing

Data-Driven Parsing

We approximate the Oracle $O$ using a Classifier $\text{Predict}(c)$ that predicts the next transition using Features of $c$, $\text{feats}(c)$.

PARSE($w_1, \ldots, w_n$)
1. $c \leftarrow ([w_0]_S, [w_1, \ldots, w_n]_Q,)$
2. while $Q_c \neq []$ or $|S_c| = 1$
3. $t \leftarrow \text{Predict}(w, \text{feats}(c))$
4. $c \leftarrow t(c)$
5. return $T = (w_0, w_1, \ldots, w_n, A_c)$
Transition-Based Dependency Parsing

Feature Engineering

\[ \text{[ROOT, had, little, effect]}_s \ [\text{on, financial, markets, .}]_q \]

\[ \text{ROOT Economic news had little effect on financial markets .} \]
Transition-Based Dependency Parsing

Feature Engineering

\[
\text{[ROOT, had, little, effect]}_S \ [\text{on, financial, markets, .}]_Q
\]

<table>
<thead>
<tr>
<th>name</th>
<th>feature</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[0] word</td>
<td>effect</td>
<td>(w_1)</td>
</tr>
<tr>
<td>S[0] pos</td>
<td>NN</td>
<td>(w_2)</td>
</tr>
<tr>
<td>S[1] word</td>
<td>little</td>
<td>(w_3)</td>
</tr>
<tr>
<td>S[1] pos</td>
<td>JJ</td>
<td>(w_4)</td>
</tr>
<tr>
<td>Q[0] word</td>
<td>on</td>
<td>(w_5)</td>
</tr>
<tr>
<td>Q[0] pos</td>
<td>P</td>
<td>(w_6)</td>
</tr>
<tr>
<td>Q[1] word</td>
<td>financial</td>
<td>(w_7)</td>
</tr>
<tr>
<td>Q[1] pos</td>
<td>JJ</td>
<td>(w_8)</td>
</tr>
<tr>
<td>Root(A) word</td>
<td>had</td>
<td>(w_9)</td>
</tr>
<tr>
<td>Root(A) POS</td>
<td>VB</td>
<td>(w_{10})</td>
</tr>
<tr>
<td>s[0]-S[1]</td>
<td>effect (\rightarrow) little</td>
<td>(w_{11})</td>
</tr>
<tr>
<td>s[1]-S[0]</td>
<td>little (\rightarrow) effect</td>
<td>(w_{12})</td>
</tr>
</tbody>
</table>
An Oracle $O$ can be approximated by a (linear) classifier:

$$\text{Predict}(t) = \arg \max_t w \Phi(c, t)$$

History-Based Features $\Phi(c, t)$
- Features over input words relative to $S$ and $Q$
- Features over the (partial) dependency tree defined by $A$
- Features over the (partial) transition sequence so far

Learning $w$ from Treebank Data
- Reconstruct Oracle sequence for each sentence
- Construct training data set $D = \{ (c, t) | O(c) = t \}$
- Maximize accuracy of local predictions $O(c) = t$
Transition-Based Dependency Parsing

Online Learning
Online learning, Perceptron

Training data: $\mathcal{T} = \{(x_t, y_t)\}_{t=1}^{\mathcal{T}}$

1. $w = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $y' = \arg \max_y w \cdot f(x_t, y)$
5. if $y' \neq y_t$
6. $w = w + f(x_t, y_t) - f(x_t, y')$
7. return $w$

Step 4: Greedy Decoding
Greedy: At each step, select the maximum scoring transition.
Reflections on Dependency Parsing

The Objective Function:

\[ t^* = \arg\max_{t \in GEN(x)} w^T \Phi(t) \]

The Modeling Choices:

<table>
<thead>
<tr>
<th></th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>Dependency Trees</td>
<td>Dependency Trees</td>
</tr>
<tr>
<td>Model</td>
<td>Discriminative</td>
<td>Discriminative</td>
</tr>
<tr>
<td></td>
<td>Transition-Based</td>
<td>Graph-Based MST</td>
</tr>
<tr>
<td>Decoder</td>
<td>Greedy Linear</td>
<td>Exhaustive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polynomial</td>
</tr>
<tr>
<td>Trainer</td>
<td>Online/Perceptron</td>
<td>Online/MIRA</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Reflections on Dependency Parsing

- CoNLL 2006 shared task [Buchholz and Marsi 2006]
  - *MaltParser* [Nivre et al. 2006] – deterministic, local learning
  - *MSTParser* [McDonald et al. 2006] – exact, global learning
  - Same average parsing accuracy over 13 languages
  - **High**: English and similar. **Low**: Morphologically rich

- Comparative error analysis [McDonald and Nivre 2007]:
  - *MaltParser* more accurate on short dependencies and disambiguation of core grammatical functions
  - *MSTParser* more accurate on long dependencies and dependencies near the root of the tree

- Hypothesized explanation for the results:
  - *MALT*: Rich features counteracted by error propagation
  - *MST*: Local features miss contextual information
  - Voting/Stacking: improves results for both
Recent Advances in Dependency Parsing

- Graph-based:
  - Larger factors instead of single arcs
- Transition-Based:
  - Beam Search instead of Linear Greedy Search
- Neural-Based:
  - Reuse and Revise Existing Architectures, OR
  - Use NMT Sequence-to-sequence with Attention
Recent Advances: Neural-Network Models (1)

The Basic Claim: Both graph based and transition-based models benefit from the move to Neural Networks.

► Same overall approach and algorithm as before, but:
  → Replace linear classifier with non-linear to MLP.
  → Use pre-trained word embeddings.
  → Replace feature-extractor with Bi-LSTM.

► Further explorations:
  → Semi-supervised learning.
  → Multi-task learning

► Remaining Challenges:
  → Out-of-domain parsing (e.g. twitter)
  → Parsing Morphologically-Rich Languages (e.g. Hebrew)
The Basic Idea: Pretend that both the sentence and the tree are sequences and use an NMT model to translate one to the other

The cat sleeps on the mat

( ( The cat ) ( sleeps ( on ( the mat ) ) ) )

More on Neural Models for sequences in DL4TEXT.
Summarising Dependency Parsing

- Dependency trees as labeled bi-lexical dependencies
  - Data-Driven parsing trained over Dependency Treebanks

- Varied Methods:
  - Conversion-Based (Rules)
  - Grammar-Based (Probabilistic)
  - Graph-Based (Linear, Globally Optimized)
  - Transition-Based (Linear, Locally Optimized)

- Neural Network models work the same but:
  - Non-linear objective *eg.* MLP
  - Better word-representations *eg.* Word Embeddings
  - Better (automatic) feature-extraction *eg.* BiLSTM
Summarising Dependency Parsing

- Dependency trees as labeled bi-lexical dependencies
  - Data-Driven parsing trained over Dependency Treebanks

- Varied Methods:
  - Conversion-Based (Rules)
  - Grammar-Based (Probabilistic)
  - Graph-Based (Linear, Globally Optimized)
  - Transition-Based (Linear, Locally Optimized)

- Neural Network models work the same but:
  - Non-linear objective *eg.* MLP
  - Better word-representations *eg.* Word Embeddings
  - Better (automatic) feature-extraction *eg.* BiLSTM

- English is “solved” — What about other languages?
  - Stanford CoreNLP https://corenlp.run
  - The UD Initiative: https://universaldependencies.org/
  - UDPipe: http://lindat.mff.cuni.cz/services/udpipe/
  - ONLP: nlp.biu.ac.il/~rtsarfaty/onlp/hebrew/
NLP@BIU: Where We’re At

So Far

✓ Part 1: Introduction (classes 1-2)
✓ Part 2: Words/Sequences (classes 3-4)
✓ Part 3: Sentences/Trees (classes 5-6)
→ Part 4: Meanings (Prof. Ido Dagan, starting class 7)

To Be Continued...