Statistical Parsing: The Big Picture

The Task

sentence → parsing system → tree
Statistical Parsing: The Big Picture

The Architecture

sentence \rightarrow \text{decoding} \rightarrow \text{tree}

\text{training} \uparrow

\text{treebank} \uparrow

\text{parsing system}
Statistical Parsing: The Big Picture

The Evaluation Framework

(sentence) → parsing system → decoding → parse-tree → evaluation metrics → scores

(sentence) → parsing system → training → parse-tree → evaluation metrics → scores

(train) → parsing system → decoding → parse-tree → evaluation metrics → scores

(test) → parsing system → decoding → parse-tree → evaluation metrics → scores

(treebank) → parsing system → decoding → parse-tree → evaluation metrics → scores

(gold-tree) → parsing system → decoding → parse-tree → evaluation metrics → scores
Statistical Parsing: The Big Picture

The Questions

- What kind of **Trees**?
- What kind of **Models**?
  - Generative
  - Discriminative
- Which Search Algorithm (**Decoding**)?
- Which Learning Algorithm (**Training**)?
- What kind of **Evaluation**?
Statistical Parsing: The Big Picture

Previously on NLP@BIU

<table>
<thead>
<tr>
<th>Representation</th>
<th>Phrase-Structure Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Generative</td>
</tr>
<tr>
<td>Objective</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Search</td>
<td>CKY</td>
</tr>
<tr>
<td>Train</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Precision/Recall/F1</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

Today: More Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Constituency Trees</th>
<th>Dependency Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Generative</td>
<td>?</td>
</tr>
<tr>
<td>Objective</td>
<td>Probabilistic</td>
<td>?</td>
</tr>
<tr>
<td>Search</td>
<td>Exhaustive</td>
<td>?</td>
</tr>
<tr>
<td>Train</td>
<td>MLE</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>F_1-Scores</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
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  PP
    V  NP  P  NP
      dumped  sacks  into  bins
Where Do Dependency Trees Come From?

S/dumped

NP/workers
  workers

NP/sacks
VP/dumped

V/dumped
dumped

PP/into

P/into
P/into

NP/bins
NP/bins
Where do Dependency Trees Come From?

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

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

-ROOT-

dumped

workers  sacks  into

  bins
Representation: Labeled vs. Unlabeled

Unlabeled Dependency Tree:

-ROOT-
  dumped
  workers sacks into

Labeled Dependency Tree:

-ROOT-
  dumped
  subj workers
d  dobj sacks
  prep into
  pobj bins
Representation: Functional vs. Lexical

Functional Dependencies:

-ROOT-

dumped

subj workers
dobj sacks
prep into
pobj bins

Lexical Dependencies:

-ROOT-

dumped

subj workers
dobj sacks
nmod bins
case into
Discussions: Options and Schemes

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

The Universal Dependencies Initiative
https://universaldependencies.org/
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!
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
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.
Evaluation Metrics

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

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

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

- **Exact Match**
  The percentage of sentences with parses identical to gold
Models for Dependency Parsing

The Parsing Objective:

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

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>Decoder</td>
<td>?</td>
</tr>
<tr>
<td>Trainer</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:

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

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

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

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

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

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

The Objective Function:

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

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>Decoder</td>
<td>?</td>
</tr>
<tr>
<td>Trainer</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

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

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>Decoder</td>
<td>?</td>
</tr>
<tr>
<td>Trainer</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

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

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>Decoder</td>
<td>?</td>
</tr>
<tr>
<td>Trainer</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Today: Introduction to Dependency Parsing

The Objective Function:

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

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>Decoder</td>
<td>?</td>
</tr>
<tr>
<td>Trainer</td>
<td>?</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Attachment Scores</td>
</tr>
</tbody>
</table>
Modeling Methods

Our Modeling Tasks:

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

- **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 \( w \)?
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

```
TOP
  S
    NP  VP
      workers
    VP  PP
      dumped  sacks
    PP
      P  NP
        into  bins
```
**Conversion-Based:** Convert PS trees Using a Head Table

<table>
<thead>
<tr>
<th></th>
<th>→</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VP</td>
<td>VBD VBN MD VBZ VB VBG VBP VP</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>NN NX JJR CD JJ JJS RB</td>
<td></td>
</tr>
<tr>
<td>ADJP</td>
<td>NNS QP NN ADVP JJ VBN VBG</td>
<td></td>
</tr>
<tr>
<td>ADVP</td>
<td>RB RBR RBS FW ADVP TO CD JJR</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>VP S SBAR ADJP UCP NP</td>
<td></td>
</tr>
<tr>
<td>SQ</td>
<td>VBZ VBD VBP VB MD PRD VP SQ</td>
<td></td>
</tr>
<tr>
<td>SBAR</td>
<td>S SQ SINV SBAR FRAG IN DT</td>
<td></td>
</tr>
</tbody>
</table>
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
Grammar-Based Dependency Parsing

The Objective Function:

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

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
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

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

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

Step 2: Defining Feature Templates
Step 2: Defining Feature Templates

<table>
<thead>
<tr>
<th>Name</th>
<th>$\phi_i$(had,OBJ, effect)</th>
<th>$w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram head</td>
<td>“had”</td>
<td>$w_{unihead}$</td>
</tr>
<tr>
<td>Unigram dep</td>
<td>“effect”</td>
<td>$w_{unidep}$</td>
</tr>
<tr>
<td>Unigram head pos</td>
<td>VB</td>
<td>$w_{uniheadpos}$</td>
</tr>
<tr>
<td>Unigram dep pos</td>
<td>NN</td>
<td>$w_{unidepos}$</td>
</tr>
<tr>
<td>Bigram head-dep</td>
<td>“had-effect”</td>
<td>$w_{bigram}$</td>
</tr>
<tr>
<td>Bigram headpos-deppos</td>
<td>VB-NN</td>
<td>$w_{bigrampos}$</td>
</tr>
<tr>
<td>Labeled Bigram head-dep</td>
<td>“had-OBJ-effect”</td>
<td>$w_{bigramlabel}$</td>
</tr>
<tr>
<td>Labeled Bigram headpos-deppos</td>
<td>VB-obj-N</td>
<td>$w_{bigramposlabel}$</td>
</tr>
<tr>
<td>In-Between pos</td>
<td>VB-IN-NN</td>
<td>$w_{inbetween}$</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. \hspace{1em} $z_i = \arg \max_{y \in \mathcal{Y}} g(y; x_i, w)$
4. \hspace{1em} gold $\leftarrow \sum \{a | y_i(a) = 1\} \phi(x_i, a)$
5. \hspace{1em} best $\leftarrow \sum \{a | z_i(a) = 1\} \phi(x_i, a)$
6. \hspace{1em} $w \leftarrow w + \text{gold} - \text{best}$
7. return $w$
Graph-Based Dependency Parsing

Step 3: Online Learning

Perceptron

- **Theory:**
  - If possible, will learn to separate the correct structure from the incorrect structures
  - Ie. 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?article=1056&context=cis_reports
Graph-Based Dependency Parsing

The Objective Function:

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

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
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:**
  \[
  ([...], [w_i, ...]Q, A) \rightarrow ([..., w_i]s, [...], Q, A)
  \]

- **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
Deterministic Parsing

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

\[
\text{PARSE}(w_1, \ldots, w_n)\\
1. \quad c \leftarrow ([w_0]_S, [w_1, \ldots, w_n]_Q)\\
2. \quad \textbf{while } Q_c \neq [] \textbf{ or } |S_c| = 1\\
3. \quad t \leftarrow O(c)\\
4. \quad c \leftarrow t(c)\\
5. \quad \textbf{return } T = (w_0, w_1, \ldots, w_n, A_c)
\]
Data-Driven Parsing

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

\[
\text{PARSE}(w_1, \ldots, w_n)
\]

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

Feature Engineering

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

\[
\begin{array}{c}
\text{ATT} \\
\text{ROOT} \quad Economic \quad news \\
\text{SBJ} \\
\text{had} \quad little \quad effect \quad on \quad financial \quad markets \quad .
\end{array}
\]
Transition-Based Dependency Parsing

Feature Engineering

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

\[
\text{ROOT} \quad \text{Economic} \quad \text{news} \quad \text{had} \quad \text{little} \quad \text{effect} \quad \text{on} \quad \text{financial} \quad \text{markets} \quad .
\]

<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>
Transition-Based Dependency Parsing

- 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 algorithms

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:

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

The Modeling Choices:

<table>
<thead>
<tr>
<th>Representation</th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency Trees</td>
<td>Dependency Trees</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discriminative</td>
<td>Discriminative</td>
<td></td>
</tr>
<tr>
<td>Transition-Based</td>
<td>Graph-Based MST</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decoder</th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>Exhaustive</td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>Polynomial</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trainer</th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online/Perceptron</td>
<td>Online/MIRA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>MALTParser</th>
<th>MSTParser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachment Scores</td>
<td>Attachment Scores</td>
<td></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
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
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
Reflections on Dependency Parsing

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: Graph-Based

Considering higher-order factors:

Recent Advances: Transition-Based

Considering more (and more diverse) structures in the beam:

http://people.sutd.edu.sg/~yue_zhang/publication.html
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

- English is “solved” — What about other languages?
  - Stanford CoreNLP
  - UD Initiative: https://universaldependencies.org/
  - UDPipe: http://lindat.mff.cuni.cz/services/udpipe/
  - ONLP: nlp.biu.ac.il/~rtsarfaty/onlp/hebrew/
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...