Dependency Parsing

With thanks to Joakim Nivre and Sandra Kübler

Dependency Grammar

- Not a coherent grammatical framework: wide range of different kinds of dependency grammar
  - just as there are wide ranges of "generative syntax"
- Different core ideas than phrase structure grammar
- We will base a lot of our discussion on [Mel'čuk(1988)]

Dependency grammar is important for those interested in CL:
- Increasing interest in dependency-based approaches to syntactic parsing in recent years (e.g., CoNLL-X shared task, 2006)

Dependency Syntax

- The basic idea:
  - Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.
- In the (translated) words of Lucien Tesnière [Tesnière(1959)]:
  - The sentence is an organized whole, the constituent elements of which are words. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. [1.3] The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. [2.1] The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence *Alfred parle [...]*, *parle* is the governor and *Alfred* the subordinate. [2.2]

Overview: constituency

(1) Small birds sing loud songs

```
    S
   / \  
  NPVP
     /\  
    Smallbirds
   / \     
  sing   NP
    /\     
  loud  songs
```

Overview: dependency

The corresponding dependency tree representations [Hudson(2000)]:

```
    obj
   /\  
 sbjnmod
   /\    
Smallbirds sing loud songs
   /\    
  sbjnmod
   /\    
  sing
   /\    
birds
|     |
songs
|     |
small
|     |
loud
```

Constituency vs. Relations

- Dependency grammar is based on relationships between words, i.e., *dependency relations*
  - A → B means A governs B or B depends on A...
  - Dependency relations can refer to syntactic properties, semantic properties, or a combination of the two
  - Relation examples: subject, object, complement, (pre-/post-)adjunct, etc.
    - Subject/Agent: *John fished.*
    - Object/Patient: *Mary hit John.*
- Phrase structure grammar is based on constituents
  - Grammatical relations are not usually seen as primitives, but as being derived from structure
**Introduction**

**Dependency Structure**

- Economic news had little effect on financial markets.

**Terminology**

- Superior
- Inferior
- Head
- Dependent
- Governor
- Modifier
- Regent
- Subordinate

**Notational Variants**

- Economic
- news
- had
- little
- effect
- on
- financial
- markets

**Phrase Structure**

- Word Grammar (WG) [Hudson(1984), Hudson(1990)]
- Functional Generative Description (FGD) [Sgall et al.(1986)]
- Dependency Unification Grammar (DUG) [Hellwig(1986), Hellwig(2003)]
- Meaning-Text Theory (MTT) [Mel’čuk(1988)]
- Functional Dependency Grammar (FDG) [Tapanainen and Järvinen(1997), Järvinen and Tapanainen(1998)]
- Topological/Extensible Dependency Grammar ([T/X]DG) [Duchier and Debusmann(2001), Debusmann et al.(2004)]

**Comparison**

- Dependency structures explicitly represent:
  - head-dependent relations (directed arcs),
  - functional categories (arc labels),
  - possibly some structural categories (parts-of-speech).
- Phrase structures explicitly represent:
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions).
- Hybrid representations may combine all elements.
Some Theoretical Issues

- Dependency structure sufficient as well as necessary?
- Mono-stratal or multi-stratal syntactic representations?
- What is the nature of lexical elements (nodes)?
  - Morphemes?
  - Word forms?
  - Multi-word units?
- What is the nature of dependency types (arc labels)?
  - Grammatical functions?
  - Semantic roles?
- What are the criteria for identifying heads and dependents?
- What are the formal properties of dependency structures?

Some Clear Cases

Construction Head Dependent
Exocentric Verb Subject (sbj)
Verb Object (obj)
Endocentric Verb Adverbial (vmod)
Noun Attribute (nmod)

Economic news suddenly affected financial markets.

Some Tricky Cases

- Complex verb groups (auxiliary ↔ main verb)
- Subordinate clauses (complementizer ↔ verb)
- Coordination (coordinator ↔ conjuncts)
- Prepositional phrases (preposition ↔ nominal)
- Punctuation

Dependency Graphs

- A dependency structure can be defined as a directed graph $G$, consisting of:
  - A set $V$ of nodes,
  - A set $E$ of arcs (edges),
  - A linear precedence order $<$ on $V$.
- Labeled graphs:
  - Nodes in $V$ are labeled with word forms (and annotation).
  - Arcs in $E$ are labeled with dependency types.
- Notational conventions $(i, j \in V)$:
  - $i \rightarrow j \equiv (i, j) \in E$
  - $i \rightarrow^* j \equiv i = j \lor \exists k : i \rightarrow k, k \rightarrow^* j$

Formal Conditions on Dependency Graphs

- $G$ is (weakly) connected:
  - For every node $i$ there is a node $j$ such that $i \rightarrow j$ or $j \rightarrow i$.
- $G$ is acyclic:
  - If $i \rightarrow j$ then not $j \rightarrow^* i$.
- $G$ obeys the single-head constraint:
  - If $i \rightarrow j$, then not $k \rightarrow j$, for any $k \neq i$.
- $G$ is projective:
  - If $i \rightarrow j$ then $i \rightarrow^* k$, for any $k$ such that $i < k < j$ or $j < k < i$.

Connectedness, Acyclicity and Single-Head

- Intuitions:
  - Syntactic structure is complete (Connectedness).
  - Syntactic structure is hierarchical (Acyclicity).
  - Every word has at most one syntactic head (Single-Head).
- Connectedness can be enforced by adding a special root node.
Introduction

Projectivity

- Most theoretical frameworks do not assume projectivity.
- Non-projective structures are needed to account for:
  - Long-distance dependencies,
  - Free word order.

What did economic news have little effect on?

Where we’re going

- Dependency parsing:
  - Input: Sentence \( x = w_1, \ldots, w_n \)
  - Output: Dependency graph \( G \)
- Focus today:
  - Computational methods for dependency parsing
  - Resources for dependency parsing (parsers, treebanks)

Parsing Methods

- Three main traditions:
  - Deterministic parsing (specifically: Transition-based parsing)
  - Dynamic programming (specifically: Graph-based parsing)
  - Constraint satisfaction (not covered today)
- Special issue:
  - Non-projective dependency parsing

Covington’s Incremental Algorithm

- Deterministic incremental parsing in \( O(n^2) \) time by trying to link each new word to each preceding one [Covington(2001)]:

\[
\text{PARSE}(x = (w_1, \ldots, w_n))
= 1 \quad \text{for } i = 1 \text{ up to } n
= 2 \quad \text{for } j = i - 1 \text{ down to } 1
= 3 \quad \text{LINK}(w_i, w_j)
\]

\[
\text{LINK}(w_i, w_j) = \begin{cases} E \leftarrow E \cup \{i, j\} & \text{if } w_j \text{ is a dependent of } w_i \\ E \leftarrow E \cup \{j, i\} & \text{if } w_i \text{ is a dependent of } w_j \\ E \leftarrow E & \text{otherwise} \end{cases}
\]

- Different conditions, such as Single-Head and Projectivity, can be incorporated into the \text{LINK} operation.

Shift-Reduce Type Algorithms

Transition-based parsing

- Data structures:
  - Stack \([\ldots, w_i]_S\) of partially processed tokens
  - Queue \([w_{i+1}, \ldots]_Q\) of remaining input tokens
- Parsing actions built from atomic actions:
  - Adding arcs \((w_i \rightarrow w_j, w_j \leftarrow w_i)\)
  - Stack and queue operations
- Left-to-right parsing in \( O(n) \) time
- Restricted to projective dependency graphs
Yamada’s Algorithm

- Three parsing actions:
  - **Shift**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r \cdots q \\
    \cdots W \cdots s & \cdots q \\
    \end{array}
    \]
  - **Left**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r, W \cdots q \\
    \cdots W, W \cdots s & \cdots q \\
    \end{array}
    \]
  - **Right**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r, W \cdots q \\
    \cdots W, W \cdots s & \cdots q \\
    \end{array}
    \]

- Algorithm variants:
  - Originally developed for Japanese (strictly head-final) with only the Shift and Right actions [Kudo and Matsumoto(2002)]
  - Adapted for English (with mixed headedness) by adding the Left action [Yamada and Matsumoto(2003)]
  - Multiple passes over the input give time complexity \(O(n^2)\)

Nivre’s Algorithm

- Four parsing actions:
  - **Shift**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r \cdots q \\
    \cdots W \cdots s & \cdots q \\
    \end{array}
    \]
  - **Reduce**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r \cdots q \\
    \cdots W \cdots s & \exists W_k : W_k \rightarrow W_q \\
    \end{array}
    \]
  - **Left-Arc**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r \cdots q \\
    \cdots W, W \cdots s & \exists W_r, W_q : W_r \rightarrow W_q \\
    \end{array}
    \]
  - **Right-Arc**
    \[
    \begin{array}{c|c|c}
    \cdots s & W_r \cdots q \\
    \cdots W, W \cdots s & \exists W_r, W_q : W_q \rightarrow W_r \\
    \end{array}
    \]

- Characteristics:
  - Integrated labeled dependency parsing
  - Arc-eager processing of right-dependents
  - Single pass over the input gives time complexity \(O(n)\)

Classifier-Based Parsing

- Data-driven deterministic parsing:
  - Deterministic parsing requires an oracle.
  - An oracle can be approximated by a classifier.
  - A classifier can be trained using treebank data.

- Learning methods:
  - Support vector machines (SVM)
    [Kudo and Matsumoto(2002), Yamada and Matsumoto(2003),
    Isozaki et al.(2004), Isozaki, Kazawa and Hirao,
    Cheng et al.(2004), Cheng, Asahara and Matsumoto,
    Nivre et al.(2006), Nivre, Hall, Nilsson, Eyirig and Marinov]
  - Memory-based learning (MBL)
    [Nivre et al.(2004), Nivre, Hall and Nilsson, Nivre and Scholz(2004)]
  - Maximum entropy modeling (MaxEnt)
    [Cheng et al.(2005), Cheng, Asahara and Matsumoto]

Comparing Algorithms

- Parsing algorithm:
  - Nivre’s algorithm gives higher accuracy than Yamada’s algorithm for parsing the Chinese CKIP treebank
    [Cheng et al.(2004), Cheng, Asahara and Matsumoto].

- Learning algorithm:
  - SVM gives higher accuracy than MaxEnt for parsing the Chinese CKIP treebank
    [Cheng et al.(2004), Cheng, Asahara and Matsumoto].
  - SVM gives higher accuracy than MBL with lexicalized feature models for three languages
    [Hall et al.(2006), Hall, Nivre and Nilsson]:
    - Chinese (Penn)
    - English (Penn)
    - Swedish (Talbanken)
**Dynamic Programming**

- Basic idea: Treat dependencies as constituents.
- Use, e.g., CYK parser (with minor modifications).
- Dependencies as constituents:

```
  nmod  sbj
  the  dog  barked  ⇒  sbj  barked
  dog

  nmod  the
```

**Dependency Chart Parsing**

- Grammar is regarded as context-free, in which each node is lexicalized.
- Chart entries are subtrees, i.e., words with all their left and right dependents.
- Problem: Different entries for different subtrees spanning a sequence of words with different heads.
- Time requirement: $O(n^5)$.

**Dynamic Programming Approaches**

- Original version: [Hays(1964)]
- Link Grammar: [Sleator and Temperley(1991)]
- Earley-style parser with left-corner filtering: [Lombardo and Lesmo(1996)]
- Bilexical grammar with discriminative estimation methods: [McDonald et al.(2005a), McDonald, Crammer and Pereira, McDonald et al.(2005b)]

**Eisner’s Bilexical Algorithm**

- Two novel aspects:
  - Modified parsing algorithm
  - Probabilistic dependency parsing
- Time requirement: $O(n^3)$.
- Modification: Instead of storing subtrees, store spans.
- Def. span: Substring such that no interior word links to any word outside the span.
- Underlying idea: In a span, only the endwords are active, i.e., still need a head.
- One or both of the endwords can be active.

**Example**

```
the man in the corner taught his dog to play golf
```

Spans:

```
( man in the corner ) ( dog to play )
```

**Assembly of Correct Parse**

Start by combining adjacent words to minimal spans:

```
( the man ) ( man in ) ( in the ) ...
```

Combine spans which overlap in one word; this word must be governed by a word in the left or right span.
Eisen's Probability Models

- Model A: Bigram lexical affinities
  - First generates a trigram Markov model for POS tagging.
  - Decides for each word pair whether they have a dependency.
  - Model is leaky because it does not control for crossing dependencies, multiple heads, ...
- Model B: Selectional preferences
  - First generates a trigram Markov model for POS tagging.
  - Each word chooses a subcat/supercat frame.
  - Selects an analysis that satisfies all frames if possible.
  - Model is also leaky because last step may fail.
- Model C: Recursive Generation
  - Each word generates its actual dependents.
  - Two Markov chains:
    - Left dependents
    - Right dependents
  - Model is not leaky.

Eisen's Results

- 25,000 Wall Street Journal sentences
- Baseline: most frequent tag chosen for a word, each word chooses a head with most common distance
- Model X: trigram tagging, no dependencies
- For comparison: state-of-the-art constituent parsing, Charniak: 92.2 F-measure

<table>
<thead>
<tr>
<th>Model</th>
<th>Non-punct</th>
<th>Tagging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>41.9</td>
<td>76.1</td>
</tr>
<tr>
<td>Model X</td>
<td>too slow</td>
<td>93.1</td>
</tr>
<tr>
<td>Model A</td>
<td>83.8</td>
<td>92.8</td>
</tr>
<tr>
<td>Model B</td>
<td>86.9</td>
<td>92.0</td>
</tr>
<tr>
<td>Model C</td>
<td>87.1</td>
<td>92.8</td>
</tr>
</tbody>
</table>

Maximum Spanning Trees

- Score of a dependency tree = sum of scores of dependencies
- Scores are independent of other dependencies.
- If scores are available, parsing can be formulated as maximum spanning tree problem.
- Two cases:
  - Projective: Use Eisner's parsing algorithm.
  - Non-projective: Use Chu-Liu-Edmonds algorithm for finding the maximum spanning tree in a directed graph [Chu and Liu(1965), Edmonds(1967)].
- Use online learning for determining weight vector \( w \): large-margin multi-class classification (MIRA)

Online Learning

Training data: \( T = (sent, deps)^T_{t=1} \)
1. \( w = 0; v = 0; i = 0; \)
2. for \( n : 1..N \)
3. for \( t : 1..T \)
4. \( w^{(i+1)} = \text{update } w^{(i)} \text{ according to } (sent, deps) \)
5. \( v = v + w^{(i+1)} \)
6. \( i = i + 1 \)
7. \( w = v / (N \cdot T) \)
MIRA

MIRA weight update:

$$\text{min} ||w(i+1) - w(i)||$$ so that

$$\text{score}(\text{sent}, \text{deps}) - \text{score}(\text{sent}, \text{deps}') \geq L(\text{deps}, \text{deps}')$$

$$\forall \text{deps}' \in dt(\text{sent})$$

- $$L(\text{deps}, \text{deps}')$$: loss function
- $$dt(\text{sent})$$: possible dependency parses for sentence

Results by McDonald et al. (2005a, 2005b)

- Unlabeled accuracy per word (W) and per sentence (S)

<table>
<thead>
<tr>
<th>Parser</th>
<th>English</th>
<th>Czech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W</td>
<td>S</td>
</tr>
<tr>
<td>x-best MIRA Eisner</td>
<td>90.9</td>
<td>37.5</td>
</tr>
<tr>
<td>best MIRA CLE</td>
<td>90.2</td>
<td>33.2</td>
</tr>
<tr>
<td>factored MIRA CLE</td>
<td>90.2</td>
<td>32.2</td>
</tr>
</tbody>
</table>

- New development (EACL 2006):
  - Scores of dependencies are not independent any more
  - Better results

Dependency Parsing

Evaluation on English

- Evaluation:
  - Penn Treebank (WSJ) converted to dependency graphs
  - Unlabeled accuracy per word (W) and per sentence (S)
    - Deterministic classifier-based parsers
      [Yamada and Matsumoto(2003), Isozaki et al.(2004)]
    - Spanning tree parsers with online training
      [McDonald et al.(2005a)]
    - Collins and Charniak parsers with same conversion

<table>
<thead>
<tr>
<th>Parser</th>
<th>W</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charniak</td>
<td>92.2</td>
<td>45.2</td>
</tr>
<tr>
<td>Collins</td>
<td>91.7</td>
<td>43.3</td>
</tr>
<tr>
<td>McDonald and Pereira</td>
<td>91.5</td>
<td>42.1</td>
</tr>
<tr>
<td>Isozaki et al.</td>
<td>91.4</td>
<td>40.7</td>
</tr>
<tr>
<td>McDonald et al.</td>
<td>91.0</td>
<td>37.5</td>
</tr>
<tr>
<td>Yamada and Matsumoto</td>
<td>90.4</td>
<td>38.4</td>
</tr>
</tbody>
</table>

Two Main Approaches

- Algorithms for non-projective dependency parsing:
  - Constraint satisfaction methods
    [Tapanainen and Järvinen(1997), Duchier and Debusmann(2001),
     Foth et al.(2004)]
  - McDonald’s spanning tree algorithm
    [McDonald et al.(2005b)]
  - Covington’s algorithm [Nivre(2006)]
- Post-processing of projective dependency graphs:
  - Pseudo-projective parsing [Nivre and Nilsson(2005)]
  - Corrective modeling [Hall and Novák(2005)]
  - Approximate non-projective parsing
    [McDonald and Pereira(2006)]

Non-Projective Dependency Parsing

- Many parsing algorithms are restricted to projective dependency graphs.
- Is this a problem?
- Statistics from CoNLL-X Shared Task [Buchholz and Marsi(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>
### Parsing Methods

#### Non-Projective Parsing Algorithms

- Complexity considerations:
  - Projective (Proj)
  - Non-projective (NonP)

<table>
<thead>
<tr>
<th>Problem/Algorithm</th>
<th>Proj</th>
<th>NonP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete grammar parsing</td>
<td>P</td>
<td>NP hard</td>
</tr>
<tr>
<td>Deterministic parsing</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>First order spanning tree</td>
<td>$O(n^3)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>$N$th order spanning tree ($N &gt; 1$)</td>
<td>$P$</td>
<td>NP hard</td>
</tr>
</tbody>
</table>

- Post-Processing

  - Two-step approach:
    1. Derive the best projective approximation of the correct (possibly) non-projective dependency graph.
    2. Improve the approximation by replacing projective arcs by (possibly) non-projective arcs.

  - Rationale:
    - Most "naturally occurring" dependency graphs are primarily projective, with only a few non-projective arcs.

  - Approaches:
    - Pseudo-projective parsing [Nivre and Nilsson(2005)]
    - Corrective modeling [Hall and Novák(2005)]
    - Approximate non-projective parsing [McDonald and Pereira(2006)]

- Evaluation on Czech

  - Evaluation:
    - Prague Dependency Treebank (PDT)
    - Unlabeled accuracy per word (W) and per sentence (S)
      - Non-projective spanning tree parsing
        - Approximate second-order non-projective spanning tree parser
          [McDonald et al.(2006)McDonald, Pereira, Ribarov and Hajic]
        - Approximate non-projective parsing on top of a second-order projective spanning tree parser
          [McDonald and Pereira(2006)]
      - Labeled deterministic second-order non-projective parsing with online learning [MIRA]
        [McDonald et al.(2006)McDonald, Lerman and Pereira]
      - Labeled deterministic pseudo-projective parsing with support vector machines

### Practical Issues

- Where to get the software?
  - Dependency parsers
  - Conversion programs for constituent-based treebanks

- Where to get the data?
  - Dependency treebanks
  - Treebanks that can be converted into dependency representation
Practical Issues

Parsers

- Trainable parsers
- Parsers with manually written grammars
- Concentrate on freely available parsers

Trainable Parsers

- Jason Eisner’s probabilistic dependency parser
  - Based on bilexical grammar
  - Contact Jason Eisner: jason@cs.jhu.edu
  - Written in LISP
- Ryan McDonald’s MSTParser
  - Based on the algorithms of [McDonald et al. (2005a); McDonald, Crammer and Pereira, McDonald et al. (2005b); McDonald, Pereira, Ribarov and Hajic]
  - URL: http://www.seas.upenn.edu/~ryantm/software/MSTParser/
  - Written in JAVA

Trainable Parsers (2)

- Joakim Nivre’s MaltParser
  - Inductive dependency parser with memory-based learning and SVMs
  - URL: http://w3.msi.vxu.se/~nivre/research/MaltParser.html
  - Executable versions are available for Solaris, Linux, Windows, and MacOS (open source version planned for fall 2006)

Parsers for Specific Languages

- Dekang Lin’s Minipar
  - Principle-based parser
  - Grammar for English
  - URL: http://www.cs.ualberta.ca/~lindek/minipar.htm
  - Executable versions for Linux, Solaris, and Windows
- Wolfgang Menzel’s CDG Parser
  - Weighted constraint dependency parser
  - Grammar for German, (English under construction)
  - Online demo: http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo
  - Download: http://nats-www.informatik.uni-hamburg.de/download

Parsers for Specific Languages (2)

- Taku Kudo’s CaboCha
  - Based on algorithms of [Kudo and Matsumoto (2002)], uses SVMs
  - URL: http://www.chasen.org/~taku/software/cabocha/
  - Web page in Japanese
- Gerold Schneider’s Pro3Gres
  - Probability-based dependency parser
  - Grammar for English
  - URL: http://www.ifi.unizh.ch/CL/gschneid/parser/
  - Written in PROLOG
- Daniel Sleator’s & Davy Temperley’s Link Grammar Parser
  - Undirected links between words
  - Grammar for English
  - URL: http://www.link.cs.cmu.edu/link/

Treebanks

- Genuine dependency treebanks
- Treebanks for which conversions to dependencies exist
- See also CoNLL-X Shared Task
  - URL: http://nextens.uvt.nl/~conll/
- Conversion strategy from constituents to dependencies
Dependency Treebanks

- Arabic: Prague Arabic Dependency Treebank
- Czech: Prague Dependency Treebank
- Danish: Danish Dependency Treebank
- Portuguese: Bosque: Floresta sintáctica
- Slovene: Slovene Dependency Treebank
- Turkish: METU-Sabanci Turkish Treebank

Dependency Treebanks (2)

- Prague Arabic Dependency Treebank
  - ca. 100,000 words
  - Available from LDC, license fee
  - (CoNLL-X shared task data, catalogue number LDC2006E01)
  - URL: http://ufal.mff.cuni.cz/padt/
- Prague Dependency Treebank
  - 1.5 million words
  - 3 layers of annotation: morphological, syntactical, tectogrammatical
  - Available from LDC, license fee
  - (CoNLL-X shared task data, catalogue number LDC2006E02)
  - URL: http://ufal.mff.cuni.cz/pdt2.0/

Dependency Treebanks (3)

- Danish Dependency Treebank
  - ca. 5,500 trees
  - Annotation based on Discontinuous Grammar [Kromann(2005)]
  - Freely downloadable
  - URL: http://www.id.cbs.dk/~mtk/treebank/
- Bosque, Floresta sintáctica
  - ca. 10,000 trees
  - Freely downloadable
  - URL: http://acdc.linguateca.pt/treebank/info_floresta_English.html

Dependency Treebanks (4)

- Slovene Dependency Treebank
  - ca. 30,000 words
  - Freely downloadable
  - URL: http://n1.ijs.si/sdt/
- METU-Sabanci Turkish Treebank
  - ca. 7,000 trees
  - Freely available, license agreement
  - URL: http://www.ii.metu.edu.tr/~corpus/treebank.html

Constituent Treebanks

- English: Penn Treebank
- Bulgarian: BulTreebank
- Chinese: Penn Chinese Treebank, Sinica Treebank
- Dutch: Alpino Treebank for Dutch
- German: TIGER/NEGRA, TüBa-D/Z
- Japanese: TüBa-J/S
- Spanish: Cast3LB
- Swedish: Talbanken05

Constituent Treebanks (2)

- Penn Treebank
  - ca. 1 million words
  - Available from LDC, license fee
  - URL: http://www.cis.upenn.edu/~treebank/home.html
  - Dependency conversion rules, available from e.g. [Collins(1999)]
  - For conversion with arc labels: Penn2Malt:
    http://v3.msi.vxu.se/~nivre/research/Penn2Malt.html
- BulTreebank
  - ca. 14,000 sentences
  - URL: http://www.bultreebank.org/
  - Dependency version available from Kiril Simov:
    kivs@bultreebank.org
### Constituent Treebanks (3)

- **Penn Chinese Treebank**
  - ca. 4000 sentences
  - Available from LDC, license fee
  - URL: [http://www.cis.upenn.edu/~chinese/ctb.html](http://www.cis.upenn.edu/~chinese/ctb.html)
  - For conversion with arc labels: Penn2Malt: [http://v3.msi.vxu.se/~nivre/research/Penn2Malt.html](http://v3.msi.vxu.se/~nivre/research/Penn2Malt.html)

- **Sinica Treebank**
  - ca. 61,000 sentences
  - Available Academia Sinica, license fee
  - URL: [http://godel.iis.sinica.edu.tw/OXIP/engversion/treebank.htm](http://godel.iis.sinica.edu.tw/OXIP/engversion/treebank.htm)
  - Dependency version available from Academia Sinica

### Constituent Treebanks (4)

- **Alpino Treebank for Dutch**
  - ca. 150,000 words
  - Freely downloadable
  - URL: [http://www.let.rug.nl/vannoostrand/trees/](http://www.let.rug.nl/vannoostrand/trees/)
  - Dependency version downloadable at [http://nextens.uvt.nl/~conll/free_data.html](http://nextens.uvt.nl/~conll/free_data.html)

- **TIGER/NEGRA**
  - ca. 50,000/20,000 sentences
  - Freely available, license agreement
  - TIGER URL: [http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/](http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/)
  - Dependency version of TIGER is included in release

### Constituent Treebanks (5)

- **TüBa-D/Z**
  - ca. 22,000 sentences
  - Freely available, license agreement
  - URL: [http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml](http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml)
  - Dependency version available from SfS Tübingen

- **TüBa-J/S**
  - Dialog data
  - ca. 18,000 sentences
  - Freely available, license agreement
  - Dependency version available from SfS Tübingen
  - URL: [http://www.sfs.uni-tuebingen.de/en_tuebajs.shtml](http://www.sfs.uni-tuebingen.de/en_tuebajs.shtml) (under construction)

### Constituent Treebanks (6)

- **Cast3LB**
  - ca. 18,000 sentences
  - URL: [http://www.dlsi.ua.es/projectes/3lb/index_en.html](http://www.dlsi.ua.es/projectes/3lb/index_en.html)
  - Dependency version available from Toni Marti (amarti@ub.edu)

- **Talbanken05**
  - ca. 300,000 words
  - Freely downloadable
  - URL: [http://v3.msi.vxu.se/~nivre/research/Talbanken05.html](http://v3.msi.vxu.se/~nivre/research/Talbanken05.html)
  - Dependency version also available

---

### References


Gaifman, Haim (1965). Dependency systems and phrase-structure systems.


