Dependency Syntax

- 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
  NP       VP
    Small birds       sing NP
      loud songs
```

Overview: dependency

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

```
obj
  nmod sbj nmod
    Small birds sing loud songs
      sing
        birds songs
          small loud
```

Economic news had little effect on financial markets.

```
Economic news had little effect on financial markets.
```
Economic news had little effect on financial markets.
Economic news had little effect on financial markets.

Superior Inferior
Head Dependent
Governor Modifier
Regent Subordinate

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

Dependency graphs 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 $(i,j \in V)$:
- $i \rightarrow j \equiv (i,j) \in E$
- $i \rightarrow^* j \equiv i = j \lor \exists k : i \rightarrow k, k \rightarrow^* j$

$G$ is (weakly) connected:
- For every node $i$ there is a node $j$ such that $i \rightarrow j$ or $j \rightarrow i$.

$G$ is acyclic:
- $i \rightarrow j$ then not $j \rightarrow^* i$.

$G$ obeys the single-head constraint:
- $i \rightarrow j$, then not $k \rightarrow j$, for any $k \neq i$.

$G$ is projective:
- $i \rightarrow j$ then $i \rightarrow^* k$, for any $k$ such that $i < k < j$ or $j < k < i$. 
Introduction

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.

Economic news had little effect on financial markets.

Where we're going

- Dependency parsing:
  - Input: Sentence $x = w_1, \ldots, w_n$
  - Output: Dependency graph $G$
- Focus:
  - 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

Deterministic Parsing

- Basic idea:
  - Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions
  - Sometimes combined with backtracking or repair
- Motivation:
  - Psycholinguistic modeling
  - Efficiency
  - Simplicity
### 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)]:

  
  ```
  PARSE(x = (w_1, \ldots, w_n))
  1. for $i = 1$ up to $n$
  2. for $j = i - 1$ down to 1
  3. $LINK(w_i, w_j)$
  
  $LINK(w_i, w_j) = \begin{cases} 
  E \leftarrow E \cup \{i, j\} & \text{if } w_j \text{ is a dependent of } w_i \\
  E \leftarrow E \cup \{j, i\} & \text{if } w_i \text{ is a dependent of } w_j \\
  E \leftarrow E & \text{otherwise}
  \end{cases}$
  ```

- Different conditions, such as Single-Head and Projectivity, can be incorporated into the $LINK$ operation.

### Yamada’s Algorithm

- Three parsing actions:
  - **Shift**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \\
    \end{array}
    ```
  - **Left**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \\
    \hline
    \vdots
    \end{array}
    ```
  - **Right**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \rightarrow \vdots, w_i, w_j, \cdots \\
    \hline
    \vdots
    \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}
    \vdots
    \hline
    \cdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \\
    \hline
    \vdots
    \end{array}
    ```
  - **Reduce**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \\
    \hline
    \vdots
    \end{array}
    ```
  - **Left-Arc**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \\
    \hline
    \vdots
    \end{array}
    ```
  - **Right-Arc**
    ```
    \begin{array}{c}
    \vdots
    \hline
    \cdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \rightarrow \vdots, w_i, \cdots \\
    \hline
    \vdots
    \end{array}
    ```

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

### Example

- [root]$S_1$ [Economic news had little effect on financial markets $].Q_1$

- Shift
Example

```
[\texttt{root}] \texttt{Economic news} \hspace{1em} \texttt{had little effect on financial markets .}
```

Left-Arc_{nmod}

Example

```
[\texttt{root}] \texttt{Economic news} \hspace{1em} \texttt{had little effect on financial markets .}
```

Shift

Example

```
[\texttt{root}] \texttt{Economic news} \hspace{1em} \texttt{had little effect on financial markets .}
```

Right-Arc_{pred}

Example

```
[\texttt{root}] \texttt{Economic news} \hspace{1em} \texttt{had} \hspace{1em} \texttt{effect on financial markets .}
```

Shift

Example

```
[\texttt{root}] \texttt{Economic news} \hspace{1em} \texttt{had little} \hspace{1em} \texttt{effect on financial markets .}
```

Left-Arc_{nmod}
Example

Economic news had little effect on financial markets.

Right-Arc

Economic news had little effect on financial markets.

Right-Arc

Economic news had little effect on financial markets.

Right-Arc

Economic news had little effect on financial markets.

Right-Arc

Economic news had little effect on financial markets.

Reduce
### Classifier-Based Parsing

- **Data-driven deterministic parsing:**
  - Deterministic parsing requires an oracle.
  - An oracle can be approximated by a classifier.

- **Learning methods:**
  - Support vector machines (SVM)

  - Memory-based learning (MBL)

  - Maximum entropy modeling (MaxEnt)
    - Cheng et al. (2005), Cheng, Asahara and Matsumoto

### Feature Models

- **Learning problem:**
  - Approximate a function from parser states, represented by feature vectors to parser actions, given a training set of gold standard derivations.

- **Typical features:**
  - Tokens:
    - Target tokens
    - Linear context (neighbors in S and Q)
  - Structural context (parents, children, siblings in G)

- **Attributes:**
  - Word form (and lemma)
  - Part-of-speech (and morpho-syntactic features)
  - Dependency type (if labeled)
  - Distance (between target tokens)
Comparing Algorithms

- Parsing algorithm:
  - Nivre’s algorithm gives higher accuracy than Yamada’s algorithm for parsing the Chinese CKIP treebank

- Learning algorithm:
  - SVM gives higher accuracy than MaxEnt for parsing the Chinese CKIP treebank
  - 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)

Graph-Based Parsing

From here, we’ll look at some slides from NASSLLI 2010 ...

Parsing Methods
Comparing Algorithms

- Parsing algorithm:
  - Nivre’s algorithm gives higher accuracy than Yamada’s algorithm for parsing the Chinese CKIP treebank

- Learning algorithm:
  - SVM gives higher accuracy than MaxEnt for parsing the Chinese CKIP treebank
  - 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)