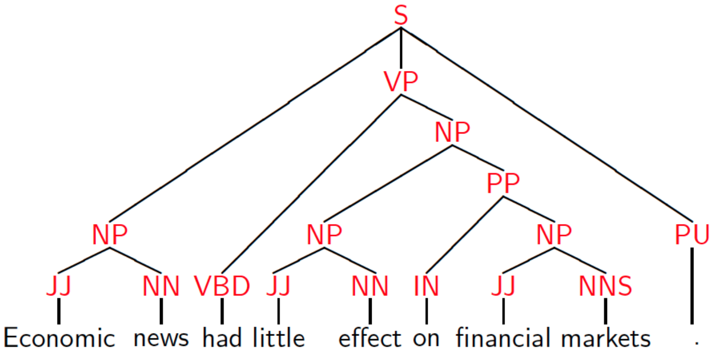
Transition Based Dependency Parsing

Pawan Goyal

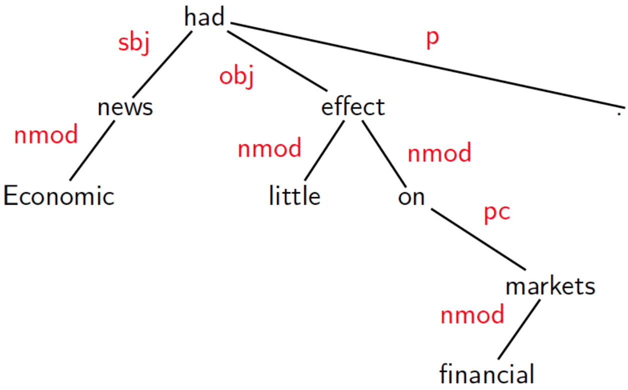
CSE, IIT Kharagpur

ACM Summer School

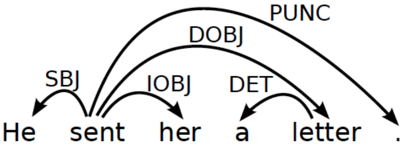
Phrase Structure



Dependency Structure Representation



Dependency Structure



- Connects the words in a sentence by putting arrows between the words.
- Arrows show relations between the words and are typed by some grammatical relations.
- Arrows connect a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate).
- Usually dependencies form a tree.

Comparison

Phrase structures explicitly represent

- Phrases (nonterminal nodes)
- Structural categories (nonterminal labels)

Comparison

Phrase structures explicitly represent

- Phrases (nonterminal nodes)
- Structural categories (nonterminal labels)

Dependency structures explicitly represent

- Head-dependent relations (directed arcs)
- Functional categories (arc labels)

Dependency Graphs

- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes,
  - a set A of arcs (edges),

Dependency Graphs

- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes,
  - a set A of arcs (edges),
- Labeled graphs:
  - Nodes in V are labeled with word forms (and annotation).
  - Arcs in A are labeled with dependency types.

Dependency Graphs

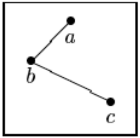
- A dependency structure can be defined as a directed graph G, consisting of
  - a set V of nodes,
  - a set A of arcs (edges),
- Labeled graphs:
  - Nodes in V are labeled with word forms (and annotation).
  - Arcs in A are labeled with dependency types.
- Notational convention:
  - Arc (w_i, d, w_j) links head w_i to dependent w_j with label d
  - $w_i \xrightarrow{d} w_j \Leftrightarrow (w_i, d, w_j) \in A$
  - $i \rightarrow j \equiv (i,j) \in A$
  - $i \rightarrow^* j \equiv i = j \vee \exists k : i \rightarrow k, k \rightarrow^* j$

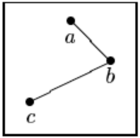
Formal conditions on Dependency Graphs

- G is 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 $j \rightarrow^* k$, for any k such that both j and k lie on the same side of i.

Formal conditions on Dependency Graphs

- G is connected:
  - For every node i there is a node j such that $i \to j$ or $j \to i$.
- G is acyclic:
  - if $i \to j$ then not $j \to^* i$.
- G obeys the single head constraint:
  - if $i \to j$ then not $k \to j$, for any $k \neq i$.
- G is projective:
  - if $i \to j$ then $j \to^* k$, for any k such that both j and k lie on the same side of i.

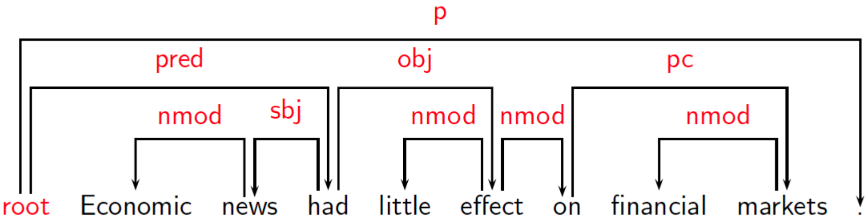




Formal Conditions: Basic Intuitions

Connectedness, Acyclicity and Single-Head

- **Connectedness:** Syntactic structure is complete.
- **Acyclicity:** Syntactic structure is hierarchical.
- **Single-Head:** Every word has at most one syntactic head.
- **Projectivity:** No crossing of dependencies.



Dependency Parsing

Dependency Parsing

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

Parsing Methods

- Deterministic Parsing
- Maximum Spanning Tree Based
- Constraint Propagation Based

Deterministic Parsing

Basic idea

Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions

Deterministic Parsing

Basic idea

Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions

Configurations

A parser configuration is a triple $c = (S, B, A)$, where

- S : a stack $[\ldots, w_i]_S$ of partially processed words,
- B : a buffer $[w_j, \ldots]_B$ of remaining input words,
- A : a set of labeled arcs (w_i, d, w_j).

Stack	**Buffer**	**Arcs**
$[\text{sent, her, a}]_S$	$[\text{letter, .}]_B$	He $\xleftarrow{\text{SBJ}}$ sent

Transition System

A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$*, where*

- C is a set of configurations,
- T is a set of transitions, such that $t : C \to C$,
- c_s is an initialization function
- $C_t \subseteq C$ is a set of terminal configurations.

Transition System

A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$, where

- C is a set of configurations,
- T is a set of transitions, such that $t : C \rightarrow C$,
- c_s is an initialization function
- $C_t \subseteq C$ is a set of terminal configurations.

A transition sequence for a sentence x is a set of configurations
$C_{0,m} = (c_o, c_1, \ldots, c_m)$ such that
$c_o = c_s(x)$, $c_m \in C_t$, $c_i = t(c_{i-1})$ for some $t \in T$

Transition System

A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$, where

- C is a set of configurations,
- T is a set of transitions, such that $t : C \rightarrow C$,
- c_s is an initialization function
- $C_t \subseteq C$ is a set of terminal configurations.

A transition sequence for a sentence x is a set of configurations
$C_{0,m} = (c_o, c_1, \ldots, c_m)$ such that
$c_o = c_s(x)$, $c_m \in C_t$, $c_i = t(c_{i-1})$ for some $t \in T$

Initialization: $([]_S, [w_1, \ldots, w_n]_B, \{\})$
Termination: $(S, []_B, A)$

Transitions for Arc-Eager Parsing

$$\text{Left-Arc}(d)\ \frac{([\ldots, w_i]_S,\ [w_j, \ldots]_B,\ A)}{([\ldots]_S,\ [w_j, \ldots]_B,\ A \cup \{(w_j, d, w_i)\})}\ \neg\text{HEAD}(w_i)$$

$$\text{Right-Arc}(d)\ \frac{([\ldots, w_i]_S,\ [w_j, \ldots]_B,\ A)}{([\ldots, w_i, w_j]_S,\ [\ldots]_B,\ A \cup \{(w_i, d, w_j)\})}$$

$$\text{Reduce}\ \frac{([\ldots, w_i]_S,\ B,\ A)}{([\ldots]_S,\ B,\ A)}\ \text{HEAD}(w_i)$$

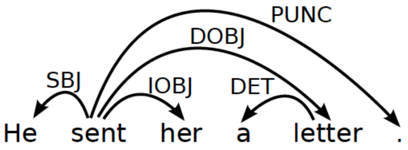
$$\text{Shift}\ \frac{([\ldots]_S,\ [w_i, \ldots]_B,\ A)}{([\ldots, w_i]_S,\ [\ldots]_B,\ A)}$$

Example

Parse Example

Transitions:

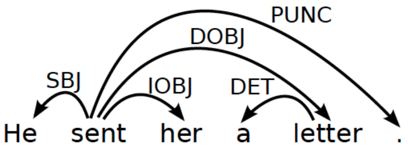
Stack	Buffer	Arcs
$[\ ]_S$	$[\text{He, sent, her, a, letter, .}]_B$	



Parse Example

Transitions: SH

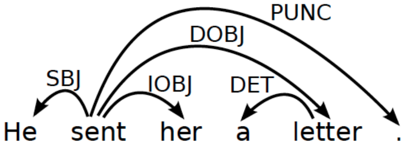
Stack	**Buffer**	**Arcs**
$[\text{He}]_S$	$[\text{sent, her, a, letter, .}]_B$	



Parse Example

Transitions: SH-LA

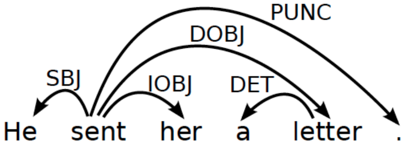
Stack	**Buffer**	**Arcs**
$[\ ]_S$	$[\text{sent, her, a, letter, .}]_B$	He $\xleftarrow{\text{SBJ}}$ sent



Parse Example

Transitions: SH-LA-SH

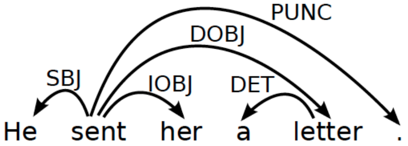
Stack	**Buffer**	**Arcs**
$[\text{sent}]_S$	$[\text{her, a, letter, .}]_B$	He $\xleftarrow{\text{SBJ}}$ sent



Parse Example

Transitions: SH-LA-SH-RA

Stack	**Buffer**	**Arcs**
$[\text{sent, her}]_S$	$[\text{a, letter, .}]_B$	He $\xleftarrow{\text{SBJ}}$ sent sent $\xrightarrow{\text{IOBJ}}$ her



Example

Parse Example

Transitions: SH-LA-SH-RA-SH

Stack

[sent, her, a]$_S$

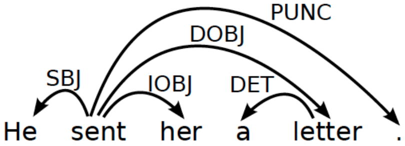
Buffer

[letter, .]$_B$

Arcs

He $\xleftarrow{\text{SBJ}}$ sent

sent $\xrightarrow{\text{IOBJ}}$ her



Parse Example

Transitions: SH-LA-SH-RA-SH-LA

Stack

$[\text{sent, her}]_S$

Buffer

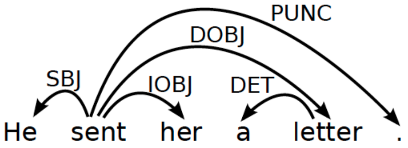
$[\text{letter, .}]_B$

Arcs

He $\xleftarrow{\text{SBJ}}$ sent

sent $\xrightarrow{\text{IOBJ}}$ her

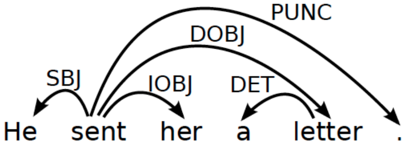
a $\xleftarrow{\text{DET}}$ letter



Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE

Stack	**Buffer**	**Arcs**
$[\text{sent}]_S$	$[\text{letter}, .]_B$	He $\xleftarrow{\text{SBJ}}$ sent sent $\xrightarrow{\text{IOBJ}}$ her a $\xleftarrow{\text{DET}}$ letter



Example

Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA

Stack

$[\text{sent, letter}]_S$

Buffer

$[.]_B$

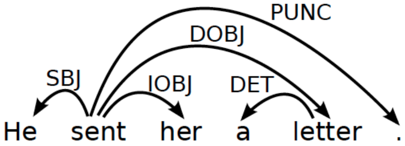
Arcs

He $\xleftarrow{\text{SBJ}}$ sent

sent $\xrightarrow{\text{IOBJ}}$ her

a $\xleftarrow{\text{DET}}$ letter

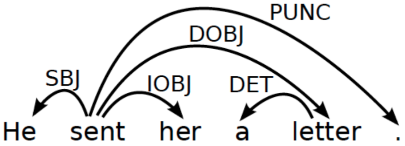
sent $\xrightarrow{\text{DOBJ}}$ letter



Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA-RE

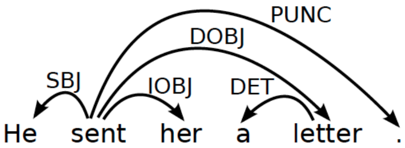
Stack	**Buffer**	**Arcs**
$[\text{sent}]_S$	$[.]_B$	$\text{He} \xleftarrow{\text{SBJ}} \text{sent}$
		$\text{sent} \xrightarrow{\text{IOBJ}} \text{her}$
		$\text{a} \xleftarrow{\text{DET}} \text{letter}$
		$\text{sent} \xrightarrow{\text{DOBJ}} \text{letter}$



Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

Stack	**Buffer**	**Arcs**
$[\text{sent}, .]_S$	$[\,]_B$	He $\xleftarrow{\text{SBJ}}$ sent
		sent $\xrightarrow{\text{IOBJ}}$ her
		a $\xleftarrow{\text{DET}}$ letter
		sent $\xrightarrow{\text{DOBJ}}$ letter
		sent $\xrightarrow{\text{PUNC}}$.



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.

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 Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.

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 Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.

Three issues

- How to represent configurations by feature vectors?
- How to derive training data from treebanks?
- How to learn classifiers?

Feature Models

A feature representation $f(c)$ of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - Structural context (parents, children, siblings in G)

Feature Models

A feature representation $f(c)$ of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - 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)

Deterministic Parsing

To guide the parser, a linear classifier can be used:

$$t^* = \arg\max_t w.f(c,t)$$

Weight vector w learned from treebank data.

Using classifier at run-time

PARSE($w_1, \ldots, w_n$)

$c \leftarrow ([]_S, [w_1, \ldots, w_n]_B, \{\})$

while $B_c \neq []$

$t^* \leftarrow \arg\max_t w.f(c,t)$

$c \leftarrow t^*(c)$

return $T = (\{w_1, \ldots, w_n\}, A_c)$

Training data

- Training instances have the form $(f(c), t)$, where
  - $f(c)$ is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., $o(c) = t$).

Training data

- Training instances have the form $(f(c), t)$, where
  - $f(c)$ is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., $o(c) = t$).
- Given a dependency treebank, we can sample the oracle function o as follows:
  - For each sentence x with gold standard dependency graph G_x, construct a transition sequence $C_{0,m} = (c_0, c_1, \dots, c_m)$ such that

    $$c_0 = c_s(x),$$
    $$G_{c_m} = G_x$$
  - For each configuration $c_i (i < m)$, we construct a training instance $(f(c_i), t_i)$, where $t_i(c_i) = c_{i+1}$.

Standard Oracle for Arc-Eager Parsing

$o(c, T) =$

- **Left-Arc** if top(S_c) $\leftarrow$ first(B_c) in T
- **Right-Arc** if top(S_c) $\rightarrow$ first(B_c) in T
- **Reduce** if $\exists w < top(S_c)$: $w \leftrightarrow$ first(B_c) in T
- **Shift** otherwise

Online Learning with an Oracle

LEARN($\{T_1, \ldots, T_N\}$)

$w \leftarrow 0.0$
for i in $1..K$
  for j in $1..N$
    $c \leftarrow ([\,]_S, [w_1, \ldots, w_{n_j}]_B, \{\})$
    while $B_c \neq [\,]$
      $t^* \leftarrow \arg\max_t w.f(c,t)$
      $t_o \leftarrow o(c, T_i)$
      if $t^* \neq t_o$
        $w \leftarrow w + f(c,t_o) - f(c,t^*)$
      $c \leftarrow t_o(c)$
return w

Oracle $o(c, T_i)$ returns the optimal transition of c and T_i

Example

Consider the sentence, 'John saw Mary'.

- Draw a dependency graph for this sentence.
- Assume that you are learning a classifier for the data-driven deterministic parsing and the above sentence is a gold-standard parse in your training data. You are also given that *John* and *Mary* are 'Nouns', while the POS tag of *saw* is 'Verb'. Assume that your features correspond to the following conditions:
  - The stack is empty
  - Top of stack is Noun and Top of buffer is Verb
  - Top of stack is Verb and Top of buffer is Noun

  Initialize the weights of all your features to *5.0*, except that in all of the above cases, you give a weight of *5.5* to *Left-Arc*. Define your feature vector and the initial weight vector.
- Use this gold standard parse during online learning and report the weights after completing one full iteration of Arc-Eager parsing over this sentence.