Dependency parsing

(based on paper written by Joakim Nivre)

Michal Mrnuštík

Fakulta informačních technologií

7. prosince 2010

Motivation

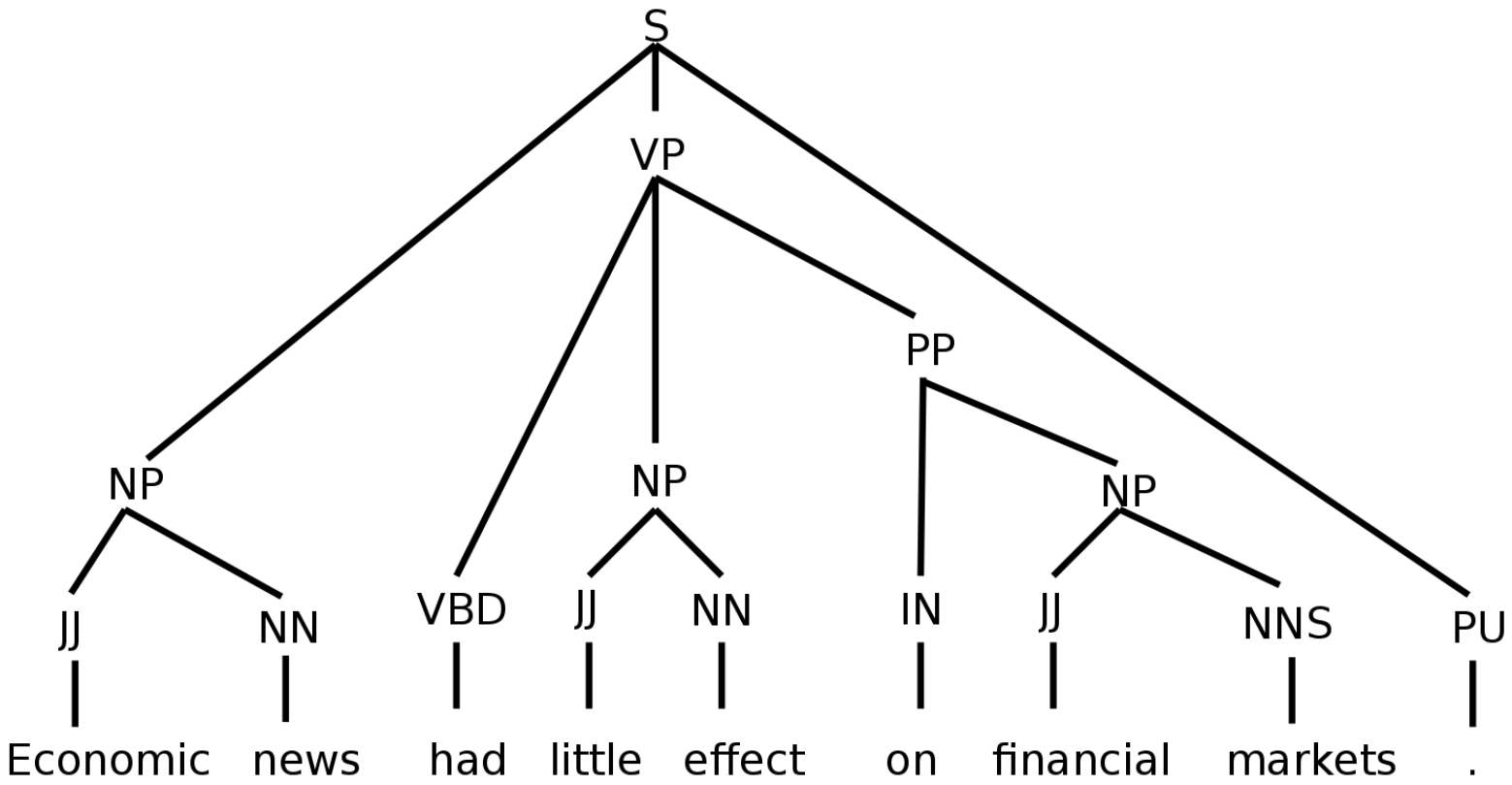
Applications

- Machine translation
- Information extraction

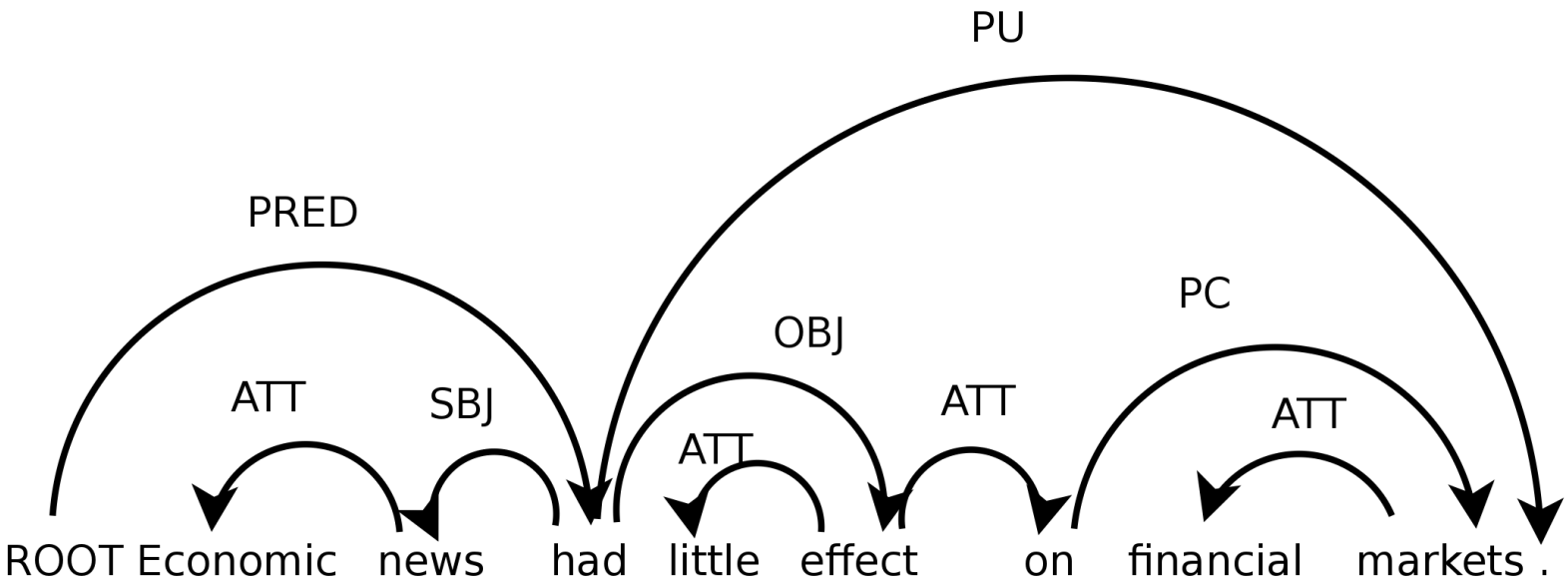
Advantages

- Transparent predicate-argument structure
- Better for languages with free or flexible word order

Phrase structure



Dependency structure



Dependency Graphs and Trees

Sentence x

- $x = w_0 w_1 \ldots w_n$
- n words $w_1 \ldots w_n$
- artificial root node w_0

Dependency graph

A dependency graph for sentence x is directed graph $G = (Vx, A)$ where:

1. $V_x = 0, 1, \ldots, n$ is a set of nodes
2. $A \subseteq V_x \times L \times V_x$ is a set of labeled arcs

Well-formed dependency graph

- tree rooted at the node 0

Approaches

Data-driven

- Transition-based
- Graph-based

Grammar-based

- Context-free
- Constraint-based

Transition System

$$S = (C, T, c_s, C_t)$$

- C is a set of *configurations*
- T is a set of *transitions*
  transition is function $t : C \rightarrow C$
- c_s is an *initialization function*, mapping a sentence $x = w_1, \ldots, w_n$ to a configuration $c \in C$.
- $C_t \subseteq C$ is set of terminal configurations.

Configurations

Configuration

$$c = (\Sigma, B, A)$$

- $c \in C$
- Σ is stack containing nodes from V_x
- B is buffer containing nodes from V_x
- A is set of dependency arcs

Configurations

Initial configuration

$$c_s(x) = ([0], [1, \ldots, n], \{\})$$

Terminal configurations

$$c_t = ([0], [], A)$$
$$c_t \in C_t \text{ for any arc set } A$$

Configurations

Configuration

$$c = ([\sigma|i], [j|\beta], A)$$

- i is node on top of the stack Σ
- σ is the rest of the stack Σ
- j is the first node in the buffer B
- β is the rest of the buffer B

Transitions

Transition sequence

$$C_{0,m} = (c_0, c_1, \ldots, c_m)$$

- $c_0 = c_s(x)$
- $c_m \in c_t$
- for every $i (1 \leq i \leq m)$, $c_i = t(c_{i-1})$ for some $t \in T$.

Result of parsing

$$G_{c_m} = (V_x, A_{c_m})$$

- G_{c_m} is dependency graph
- A_{c_m} is is the set of arcs in c_m

Transition set for projective sentences

$LeftArc_l$

$$([\sigma|i,j],B,A) \Rightarrow ([\sigma|j],B,A \cup \{(j,l,i)\})$$

$$i \neq 0$$

$RightArc_l$

$$([\sigma|i,j],B,A) \Rightarrow ([\sigma|i],B,A \cup \{(i,l,j)\})$$

Shift

$$([\sigma],[i|\beta],A) \Rightarrow ([\sigma|i],[\beta],A)$$

Transition set for projective sentences

Set of all transitions T_p

- L is set of labels
- $|T_p| = 2 \cdot |L| + 1$
- there are two transitions for each $l \in L$ ($LeftArc_l$ a $RightArc_l$) and *Shift* transition

Deterministic Transition-Based Parsing

Oracle

$$o : C \rightarrow T$$

- ideally should allways return the optimal transition t for a given configuration c
- can be aproximated by a classifier trained on treebank data

Parsing algorithm

PARSE(o,x)

1. $c \leftarrow c_s(x)$
2. **while** $c \notin C_t$
   - $t \leftarrow o(c); c \leftarrow t(c)$
3. **return** G_c

Projectivity

Projectivite arc

An arc $(w_i, l, w_j) \in A$ is projective if an only if $w_i \rightarrow^* w_k$ for all $i < k < j$ when $i < j$, or $j < k < i$ when $j < i$.

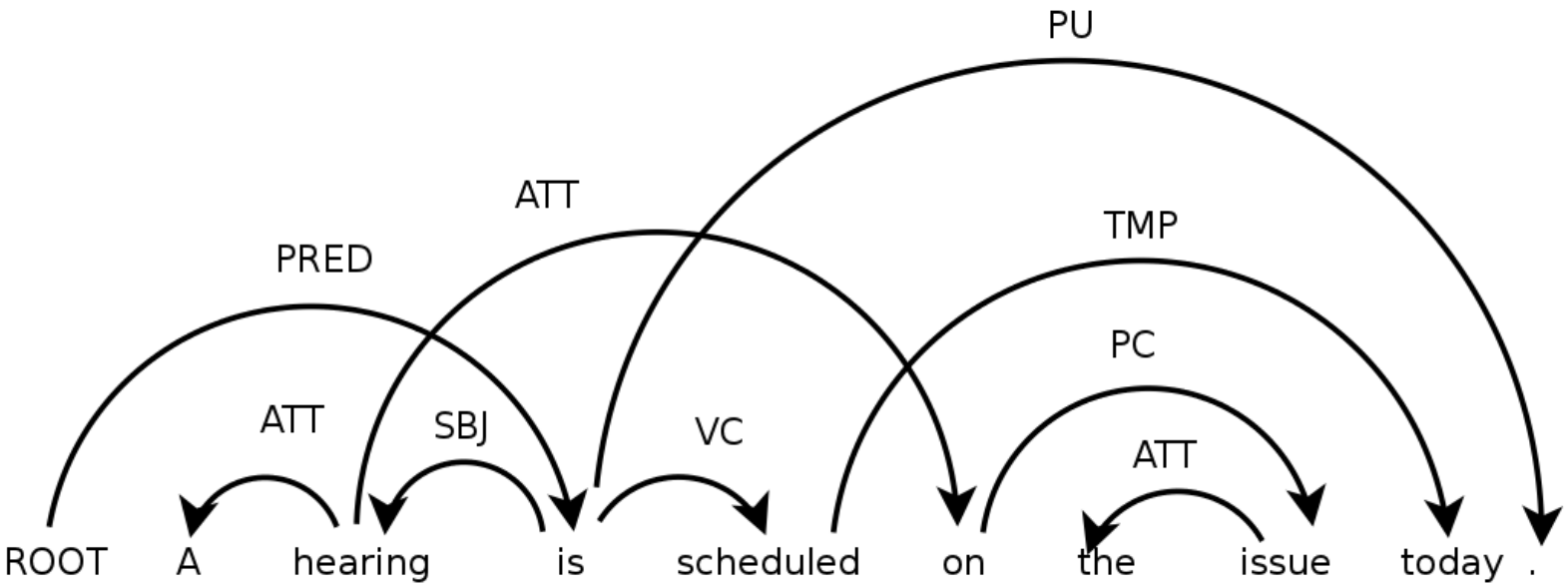
Projectivite dependency tree

A dependency tree $G = (V, A)$ is projective dependency tree if it is dependency tree and all $(w_i, l, w_j) \in A$ are projective.

Non-projectivite dependency tree

A dependency tree is non-projective dependency tree if at least one $(w_i, l, w_j) \in A$ is not projective.

Non-projective dependency structure



Transition set for non-projective sentences

Set of all transitions T_u

- All transitions for projective sentences are used.
- Swap transition is added.

$$T_u = T_p \cup \{Swap\}$$

Swap

$$([\sigma|i,j],[\beta],A) \Rightarrow ([\sigma|j],[i|\beta],A)$$

$$0 < i < j$$

- projectivity is a property of dependency tree and word order
- *Swap* and *Shift* can sort input to projective order
  1. do k *Shift* transitions to get next word in projective order to stack
  2. do $k-1$ *Swap* transitions to move preceeding words back to buffer

Thank you for your attention.

Bibliography I

- Holan, T.: Závislostní analýza češtiny.
  URL <http://ksvi.mff.cuni.cz/~holan/gram>
- Kübler, S.; McDonald, R.; Nivre, J.: Dependency Parsing.
  Synthesis Lectures on Human Language Technologies, 2009.
- Nivre, J.: Non-projective dependency parsing in expected linear time. 2009.
- Zeman, D.: A Complete Guide to Czech Language Parsing.
  URL <http://ufal.mff.cuni.cz/czech-parsing/>