Syntactic Formalisms for Parsing Natural Languages

Aleš Horák, Miloš Jakubíček, Vojtěch Kovář
(based on slides by Juyeon Kang)

ia161@nlp.fi.muni.cz

Autumn 2013

Dependency Syntax and Parsing

Outline

Motivation

- what you have seen as far: applying analysis of formal languages to a natural language – creating a phrase-structure derivation tree according to some grammar
- PS accounts for one important syntactic property: **constituency**
- is that all?
- but what about: discontinuous phrases, structure sharing

Motivation

- another crucial syntactic phenomenon is **dependency**
- what is a dependency? ”some relation between two words“
- what is the difference to phrase-structure?
- what does constituency express?
- what does dependency express?

Dependency Syntax (Mel'chuk 1988)

A more formal account – what is a dependency? A relation!

Dependency Relation

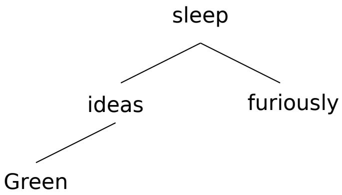
Let W be a set of all words within a sentence, then dependency relation $\rightarrow$ is $D \subseteq W \times W$ such that:

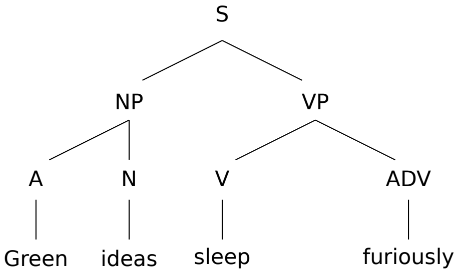
- D is **anti-reflexive**: $a \rightarrow b \Rightarrow a \neq b$
- D is **anti-symmetric**: $a \rightarrow b \wedge b \rightarrow a \Rightarrow a = b, \equiv$ (anti-reflexivity) $a \rightarrow b \Rightarrow b \nrightarrow a$
- D is **anti-transitive**: $a \rightarrow b \wedge b \rightarrow c \Rightarrow a \nrightarrow c$
- optionally: D is **labeled**: there is a mapping $l : D \rightarrow L, L$ being the set of labels

Dependency Representation

- $a \rightarrow b$: a depends on *b*, *a* is a dependent *b*, *b* is the head of a
- a **dependency graph**
- a **dependency tree**

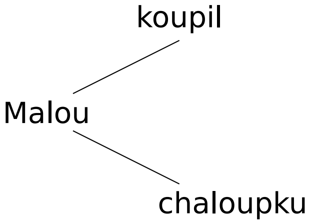
Dependency Tree vs. PS Tree





Non-projectivity

- a property of a dependency tree: a sentence is non-projective whenever drawing (projecting) a line from a node to the surface of the tree crosses an arc
- a lot of attention has been paid to this problem
- practical implications are rather limited (in most cases non-projectivity can be easily handled or avoided)
- hard cases:



Czech Tradition of Dependency Syntax

- a long tradition of dependency syntax in the Prague linguistic school (Sgall, Hajičová, Panevová)
- Institute of Formal and Applied Linguistics at Charles University
- formalized as Functional Generative Description (FGD) of language
- Prague Dependency Treebank (PDT)

Dependencies vs. PS

- is one of the formalisms clearly better than the other one? **No.**
  - dependencies: $\oplus$ account for relational phenomena, $\oplus$ simple
  - phrase-structure: $\oplus$ account for constituency, $\oplus$ easy chunking
- can we perform transformation from one of the formalism to the other one a vice versa? **Technically yes, but . . .**
  - It is not a problem to convert the structure between a dependency tree and a PS tree ...
  - ... but it is a problem to transform the information included
- $\Rightarrow$ both of the formalisms are convertible but not mutually equivalent

Dependency Parsing

- rule-based vs. statistical
- transition-based ($\rightarrow$ deterministic parsing)
- graph-based ($\rightarrow$ spanning trees algorithms)
- various other approaches (ILP, PS conversion, . . .)
- very recent advances (vs. long studied PS parsing algorithms)

Introduction to Dependency parsing

Motivation

a. dependency-based syntactic representation seem to be useful in many applications of language technology: machine translation, information extraction

$\rightarrow$ transparent encoding of predicate-argument structure

b. dependency grammar is better suited than phrase structure grammar for language with free or flexible word order

$\rightarrow$ analysis of diverse languages within a common framework

c. leading to the development of accurate syntactic parsers for a number of languages

$\rightarrow$ combination with machine learning from syntactically annotated corpora (e.g. treebank)

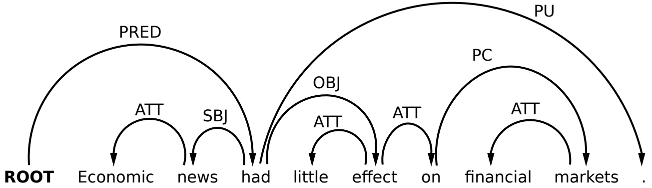
Introduction to Dependency parsing

Dependency parsing

"Task of automatically analyzing the dependency structure of a given input sentence"

Dependency parser

"Task of producing a labeled dependency structure of the kind depicted in the follow figure, where the words of the sentence are connected by typed dependency relations"



Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 1.: A sentence is a sequence of tokens denoted by

$$S = w_0 w_1 \ldots w_n$$

Def. 2.: Let $R = \{r_1, \ldots, r_m\}$ be a finite set of *possible dependency relation types* that can hold between any two words in a sentence. A relation type $r \in R$ is additionally called an *arc label*.

Definitions of dependency graphs and dependency parsing

Dependency graphs: syntactic structures over sentences

Def. 3.: A dependency graph $G = (V, A)$ is a labeled directed graph, consists of nodes, V, and arcs, A, such that for sentence $S = w_0 w_1 \dots w_n$ and label set R the following holds:

1. $V \subseteq \{w_0 w_1 \dots w_n\}$
2. $A \subseteq V \times R \times V$
3. if $(w_i, r, w_j) \in A$ then $(w_i, r', w_j) \notin A$ for all $r' \neq r$

Approach to dependency parsing

a. **data-driven**
   it makes essential use of machine learning from linguistic data in order to parse new sentences

b. **grammar-based**
   it relies on a formal grammar, defining a formal language, so that it makes sense to ask whether a given input is in the language defined by the grammar or not.

$\rightarrow$ **Data-driven have attracted the most attention in recent years.**

Data-driven approach

according to the *type of parsing model* adopted,
the algorithms used to learn the model from data
the algorithms used to parse new sentences with the model

a. **transition-based**
start by defining a transition system, or state machine, for mapping a sentence to its dependency graph.

b. **graph-based**
start by defining a space of candidate dependency graphs for a sentence.

Data-driven approach

a. **transition-based**
- **learning problem:** induce a model for predicting the next state transition, given the transition history
- **parsing problem:** construct the optimal transition sequence for the input sentence, given induced model

b. **graph-based**
- **learning problem:** induce a model for assigning scores to the candidate dependency graphs for a sentence
- **parsing problem:** find the highest-scoring dependency graph for the input sentence, given induced model

Transition-based Parsing

- Transition system consists of a set C of parser configurations and of a set D of transitions between configurations.
- **Main idea:** a sequence of valid transitions, starting in the *initial configuration* for a given sentence and ending in one of several *terminal configurations*, defines a valid dependency tree for the input sentence.

$$D_{1'm} = d_1(c_1), \ldots, d_m(c_m)$$

Transition-based Parsing

- **Definition**
  Score of $D_{1'm}$ factors by configuration-transition pairs (c_i, d_i):

  $$s(D_{1'm}) = \sum_{i=1}^{m} s(c_i, d_i)$$

- **Learning**
  Scoring function $s(c_i, d_i)$ for $d_i(c_i) \in D_{1'm}$

- **Inference**
  Search for highest scoring sequence $D^*_{1'm}$ given $s(c_i, d_i)$

Transition-based Parsing

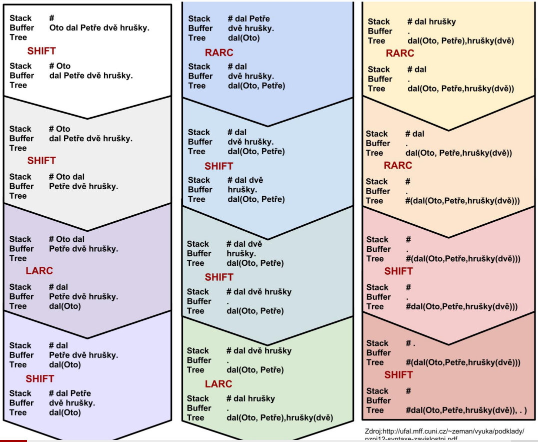
Inference for transition-based parsing

- **Common inference strategies:**
  - Deterministic [Yamada and Matsumoto 2003, Nivre et al. 2004]
  - Beam search [Johansson and Nugues 2006, Titov and Henderson 2007]
  - Complexity given by upper bound on transition sequence length
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]

Transition-based Parsing – Nivre algorithm

Prechody (*transitions*) z jedného stavu (konfigurácie) do druhého.
Konfigurácia:
- Vstupní buffer (slova vo vete zľava doprava)
- Zásobník
- Výstupní strom (slova, závislosti a značky závislostí)

Prechody:
- **Reduce**: uvolnenie vrchného slovo na zásobníku
- **Shift**: presunutie slovo z bufferu na zásobník
- **Left-arc (LARC)**: ľavá závislosť medzi dvoma hornými slovami v zásobníku
- **Right-arc (RARC)**: pravá závislosť medzi dvoma hornými slovami v zásobníku



Zdroj:http://ufal.mff.cuni.cz/~zeman/vyuka/podklady/

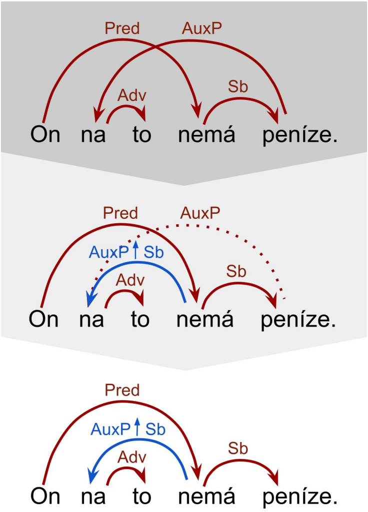
Transition-based Parsing

Learning for transition-based parsing

- **Typical scoring function:**
  - $s(c_i, d_i) = w \cdot f(c_i, d_i)$ where $f(c_i, d_i)$ is a feature vector over configuration c_i and transition d_i and w is a weight vector $[w_i = \text{ weight of feature} f_i(c_i, d_i)]$
- **Transition system**
  - Projective O(n) [Yamada and Matsumoto 2003, Nivre 2003]
  - Limited non-projective O(n) [Attardi 2006, Nivre 2007]
  - Unrestricted non-projective O(n2) [Nivre 2008, Nivre 2009]
- **Problem**
  - Learning is local but features are based on the global history

Transition-based Parsing

Projectivization to pseudo-projectivity:



Graph-based Parsing

- For a input sentence S we define a graph $G_s = (V_s, A_s)$ where
  $V_s = \{w_0, w_1, \ldots, w_n\}$ and
  $A_s = \{(w_i, w_j, l) | w_i, w_j \in V \text{ and } l \in L\}$
- Score of a dependency tree T factors by subgraphs $G_s, \ldots, Gs$:

$$s(T) = \sum_{i-1}^{m} s(G_i)$$

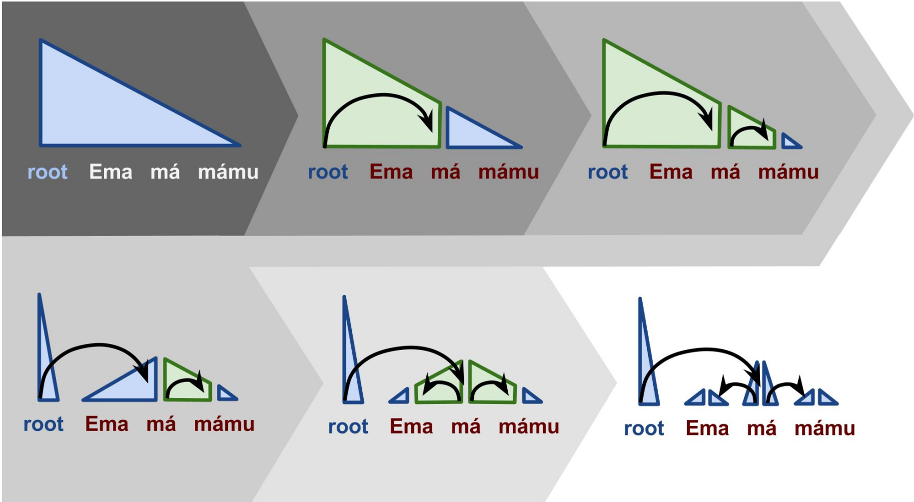
- Learning: **Scoring function** $s(G_i)$ for a subgraph $G_i \in T$
- Inference: Search for maximum spanning tree scoring sequence T^* of G_s given $s(G_i)$

Graph-based Parsing

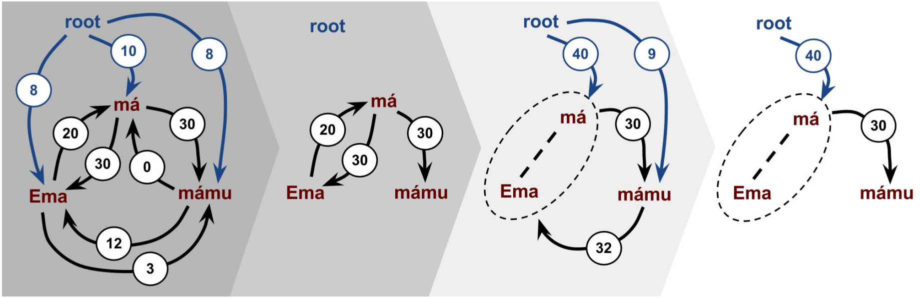
Learning graph-based models

- **Typical scoring function:**
  - $s(G_i) = w \cdot f(G_i)$ where $f(G_i)$ is a high-dimensional feature vector over subgraphs and w is a weight vector
    $[w_j =$ weight of feature $f_j(G_i)]$
- **Structured learning** [McDonald et al. 2005a, Smith and Johnson 2007]:
  - Learn weights that maximize the score of the correct dependency tree for every sentence in the training set
- **Problem**
  - Learning is global (trees) but features are local (subgraphs)

Graph-based Parsing – Eisner algorithm



Graph-based Parsing – Chu-Liu-Edmonds algorithm



Grammar-based approach

a. **context-free dependency parsing**
exploits a mapping from dependency structures to CFG structure representations and reuses parsing algorithms originally developed for CFG → chart parsing algorithms

b. **constraint-based dependency parsing**

- parsing viewed as a constraint satisfaction problem
- grammar defined as a set of constraints on well-formed dependency graphs
- finding a dependency graph for a sentence that satisfies all the constraints of the grammar (having the best score)

Grammar-based approach

a. **context-free dependency parsing**

Advantage: Well-studied parsing algorithms such as CKY, Earley's algorithm can be used for dependency parsing as well.

$\rightarrow$ need to convert dependency grammars into efficiently parsable context-free grammars; (e.g. bilexical CFG, Eisner and Smith, 2005)

b. **constraint-based dependency parsing**

defines the problem as constraint satisfaction

- Weighted constraint dependency grammar (WCDG, Foth and Menzel, 2005)
- Transformation-based CDG

Conclusions

1. Dependency syntax vs. constituency (phrase-structure) syntax
2. Non-projectivity
3. Graph-based and Transition-based methods