Natural Language Processing

Introduction to dependency parsing

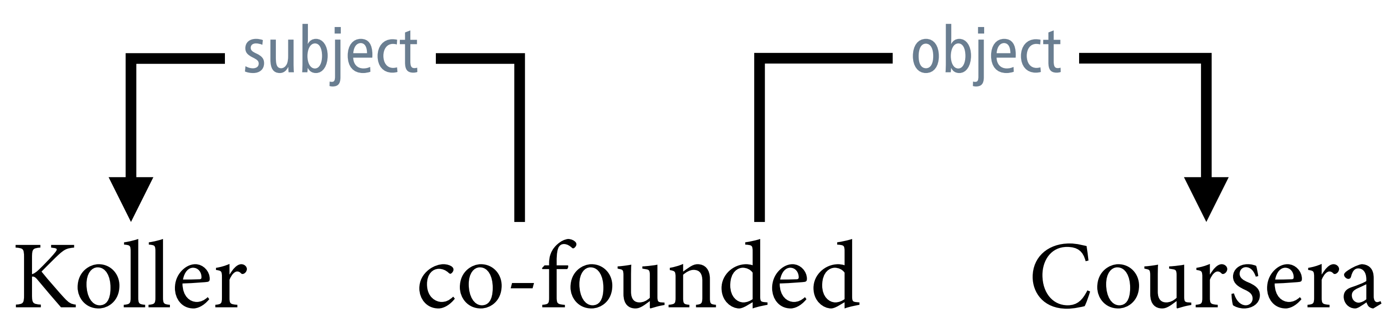
Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

Dependency parsing

- **Syntactic parsing** is the task of mapping a sentence to a formal representation of its syntactic structure.

- We focus on representations in the form of **dependency trees**.



- A syntactic dependency is an asymmetric relation between a **head** and a **dependent**.

universaldependencies.org

Current UD Languages

Information about language families (and genera for families with multiple branches) is mostly taken from [WALS Online](#) (IE = Indo-European).

Afrikaans	1	49K		IE, Germanic
Akkadian	1	1K		Afro-Asiatic, Semitic
Albanian	1	<1K	W	IE, Albanian
Amharic	1	10K		Afro-Asiatic, Semitic
Ancient Greek	2	416K		IE, Greek
Arabic	3	1,042K	W	Afro-Asiatic, Semitic
Armenian	1	52K		IE, Armenian
Assyrian	1	<1K		Afro-Asiatic, Semitic
Bambara	1	13K		Mande
Basque	1	121K		Basque
Belarusian	1	13K		IE, Slavic
Bhojpuri	2	4K		IE, Indic
Breton	1	10K	W	IE, Celtic
Bulgarian	1	156K		IE, Slavic
Buryat	1	10K		Mongolic
Cantonese	1	13K		Sino-Tibetan
Catalan	1	531K		IE, Romance
Chinese	5	285K	W	Sino-Tibetan
Classical Chinese	1	74K		Sino-Tibetan
Coptic	1	40K		Afro-Asiatic, Egyptian
Croatian	1	199K	W	IE, Slavic
Czech	5	2,222K	W	IE, Slavic
Danish	2	100K		IE, Germanic
Dutch	2	306K	W	IE, Germanic
English	9	620K	W	IE, Germanic
Erzya	1	15K		Uralic, Mordvin
Estonian	2	465K		Uralic, Finnic
Faroese	1	10K	W	IE, Germanic
Finnish	3	377K	W	Uralic, Finnic
French	8	1,157K	W	IE, Romance
Galician	2	164K		IE, Romance
German	4	3,753K	W	IE, Germanic
Gothic	1	55K		IE, Germanic
Greek	1	63K	W	IE, Greek
Hebrew	1	161K		Afro-Asiatic, Semitic

Dependency trees

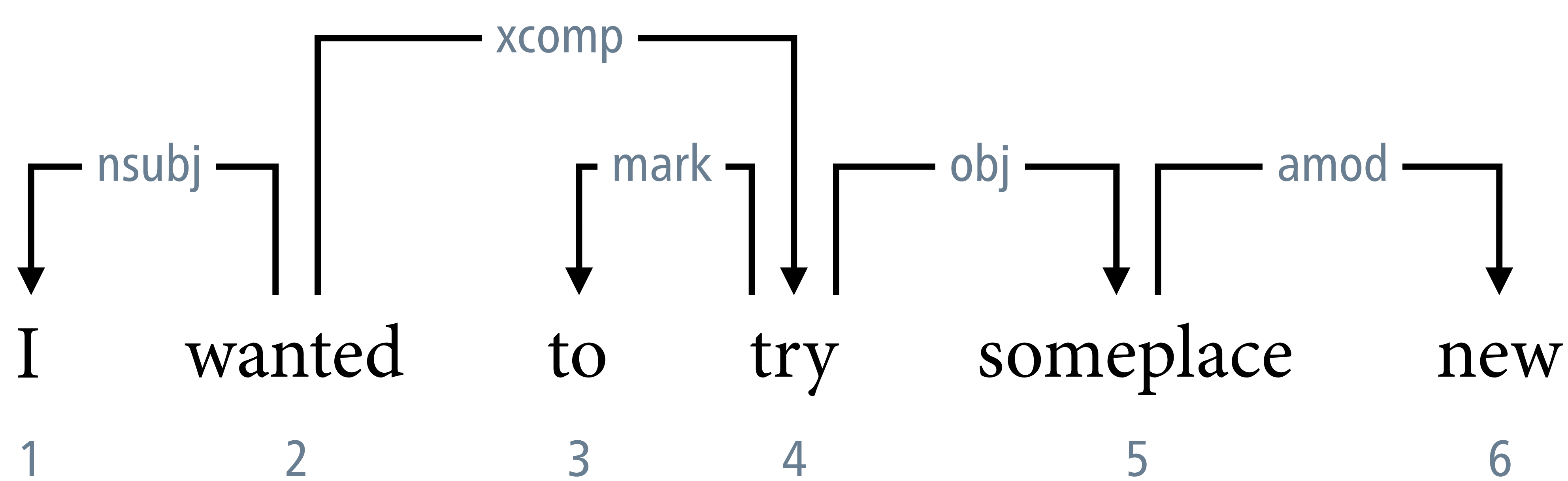
- A **dependency tree** for a sentence x is a digraph $G = (V, A)$ where $V = \{1, \ldots, |x|\}$ and where there exists a $r \in V$ such that every $v \in V$ is reachable from r via exactly one directed path.

- The vertex r is called the **root** of G.

- The arcs of a dependency tree may be labelled to indicate the type of the syntactic relation that holds between the two elements.

  Universal Dependencies v2 uses 37 universal syntactic relations (list).

Representation of dependency trees



word position	1	2	3	4	5	6
head position	2	0	4	2	4	5
dependency relation	nsubj	root	mark	xcomp	obj	amod

Two parsing paradigms

- **Graph-based dependency parsing**

  Cast parsing as a combinatorial optimisation problem over a (possibly restricted) set of dependency trees.

- **Transition-based dependency parsing**

  Cast parsing as a sequence of local classification problems: at each point in time, predict one of several parser actions.

Graph-based dependency parsing

- Given a sentence x and a set $Y(x)$ of candidate dependency trees for x, we want to find a highest-scoring tree $\hat{y} \in Y(x)$:

$$\hat{y} = \underset{y \in Y(x)}{\arg\max}\, \text{score}(x, y)$$

- The computational complexity of this problem depends on the choice of the set $Y(x)$ and the scoring function.

The arc-factored model

- Under the **arc-factored model**, the score of a dependency tree is expressed as the sum of the scores of its arcs:

$$\hat{y} = \arg\max_{y \in Y(x)} \sum_{a \in y} \text{score}(x, a)$$

head–dependent arc

- The score of a single arc can be computed by means of a neural network that receives the head and the dependent as input.

for example, a simple linear layer: $\text{score}(x, h \rightarrow d) = [\boldsymbol{h} \,;\, \boldsymbol{d}] \cdot \boldsymbol{w} + b$

Computational complexity

- Under the arc-factored model, the highest-scoring dependency tree can be found in $O(n^3)$ time (n = sentence length).

  Chu–Liu/Edmonds algorithm; McDonald et al. (2005)

- Even seemingly minor extensions of the arc-factored model entail intractable parsing.

  McDonald and Satta (2007)

- For some of these extensions, polynomial-time parsing is possible for restricted classes of dependency trees.

Transition-based dependency parsing

- We cast parsing as a sequence of local classification problems such that solving these problems builds a dependency tree.
- In most approaches, the number of classifications required for this is linear in the length of the sentence.

Transition-based dependency parsing

- The parser starts in the **initial configuration**.

  empty dependency graph

- It then calls a classifier, which predicts the **transition** that the parser should make to move to a next configuration.

  extend the partial dependency tree

- This process is repeated until the parser reaches a **terminal configuration**.

  complete dependency tree

Training transition-based dependency parsers

- To train a transition-based dependency parser, we need a treebank with gold-standard dependency trees.
- In addition to that, we need an algorithm that tells us the gold-standard transition sequence for a tree in that treebank.
- Such an algorithm is conventionally called an **oracle**.

Comparison of the two parsing paradigms

Graph-based parsing	**Transition-based parsing**
slow (in practice, cubic in the length of the sentence)	fast (quasi-linear in the length of the sentence)
restricted feature models (in practice, arc-factored)	rich feature models defined on configurations
features and weights directly defined on target structures	indirection – features and weights defined on transitions