Unsupervised Dependency Parsing

David Mareček

Institute of Formal and Applied Linguistics
Charles University in Prague

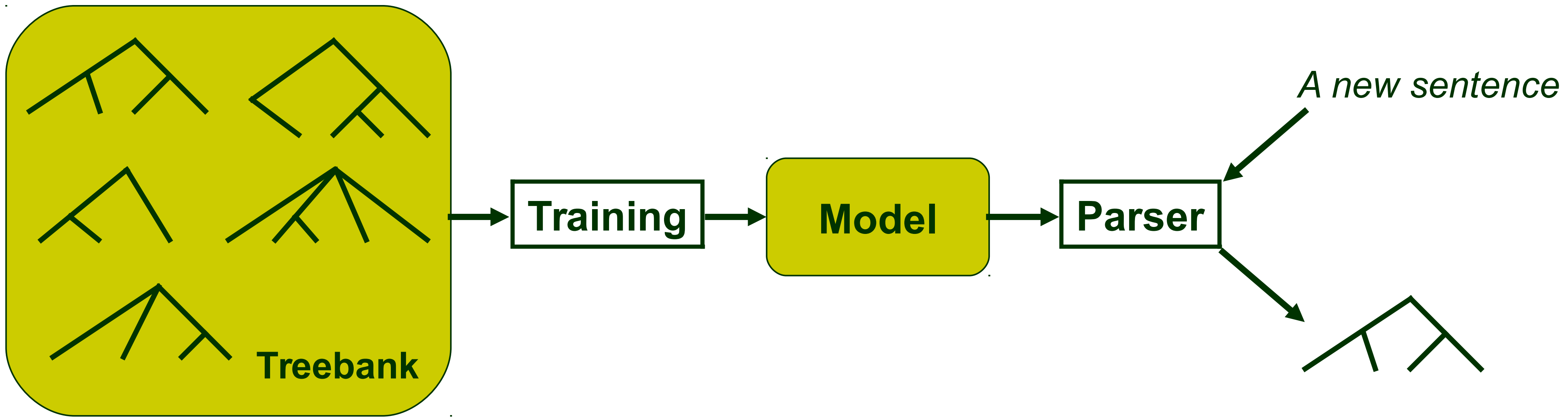
Monday seminar, ÚFAL
April 2, 2012, Prague

Outline

- What is unsupervised parsing
  - Pros & cons
  - Evaluation
- Current state-of-the-art methods
  - Dependency Model with Valence
- My work
  - Reducibility feature
  - Dependency model
  - Gibbs sampling of projective dependency trees
  - Results

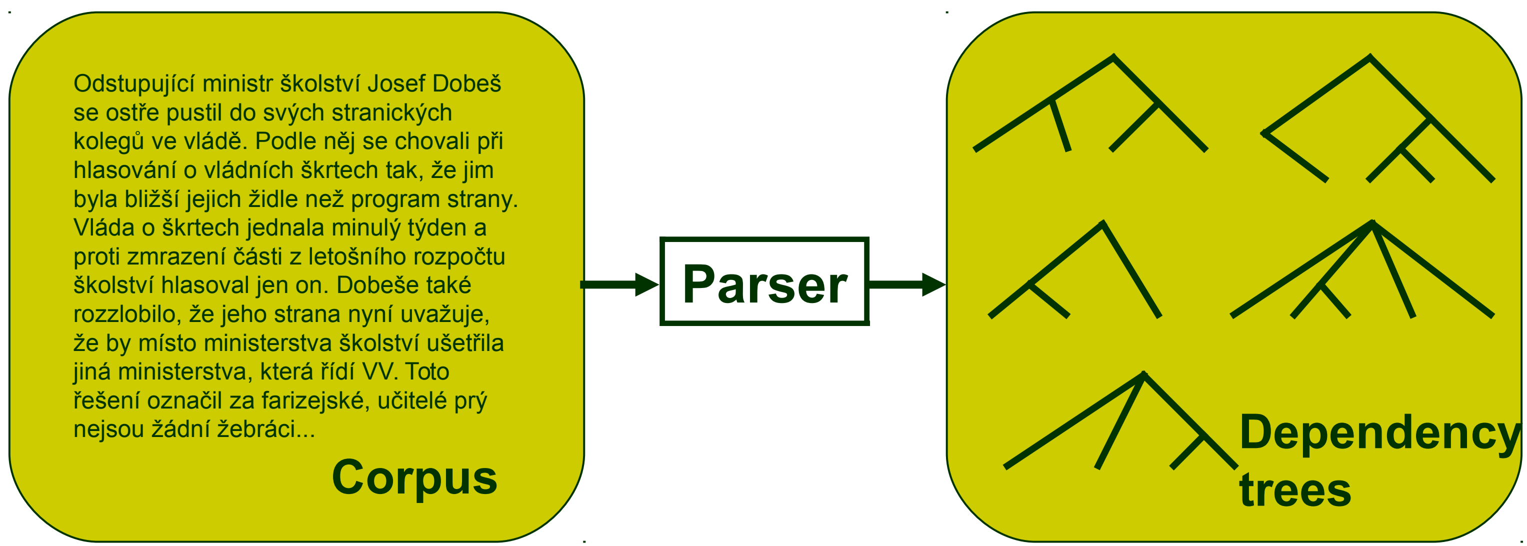
Supervised Dependency Parsing

- We have a manually annotated treebank (set of example trees), on which the parser can be learned



Unsupervised Dependency Parsing

- We have no manually annotated treebank.
- Dependency trees are induced automatically from raw (or possibly PoS tagged) texts.
- The testing data can be included into the training

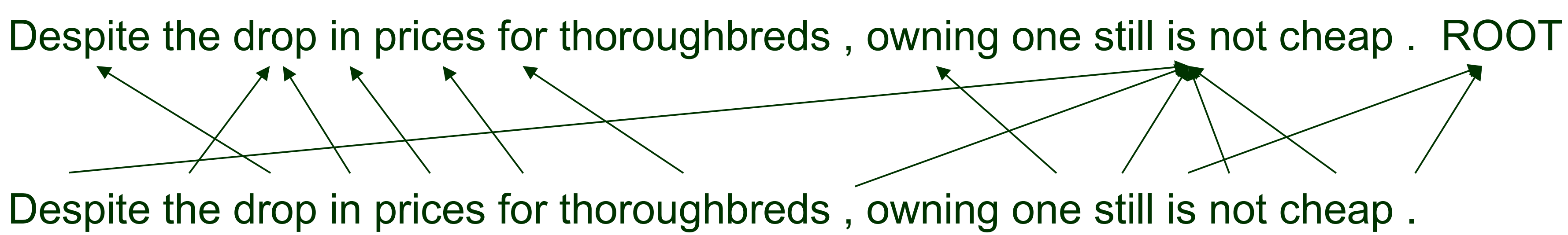


Why should be unsupervised parsing useful?

- Disadvantages:
  - So far, the results are not as good as for supervised methods (50% vs. 85% unlabeled attachment score for Czech)
- Advantages:
  - we do not need any manually annotated treebanks
  - we can possibly parse any language in any domain
  - we do not depend on tagset or tokenization used for the treebank annotation

Analogy with word-alignment

- Dependency parsing can be also seen as alignment of a sentence with itself, where
  - connecting a word to itself is disabled
  - each word is attached to just one other word (= to its parent)
  - a word can be attached to the technical root



- GIZA++ is widely used unsupervised word-alignment tool
  - easy to use
  - works on any parallel corpus and if it is large enough it achieves high quality

Evaluation metrics

- Comparison with manually annotated data is problematic
  - for each linguistic annotation, we have to make a lot of decisions how to annotate some phenomena that are not clear
  - coordination structures, auxiliary verbs, modal verbs, prepositional groups, punctuation, articles...
  - unsupervised parser can handle them differently, but, in fact, also correctly
- Two metrics:
  - UAS (unlabeled attachment score) – standard metric for evaluation of dependency parsers
  - UUAS (undirected unlabeled attachment score) – edge direction is disregarded (it is not a mistake if governor and dependent are switched)
- Ideally, the parsing quality should be measured extrinsically in some application
  - machine translation, question answering, ...
- However, the most common is the standard UAS

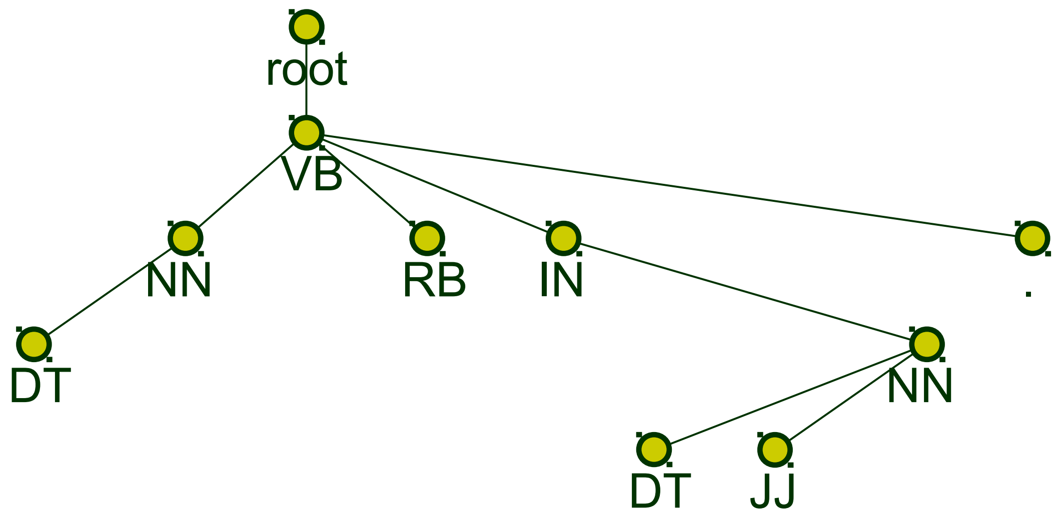
CURRENT METHODS FOR UNSUPERVISED DEPENDENCY PARSING

History of unsupervised parsing

- First approaches based on pointwise mutual information had problems in being better then right/left chain baseline
- 2005: Dan Klein introduces a Dependency Model with Valence (DMV)
  - Current state-of-the-art methods are based on modifications of DMV

Dependency Model with Valence

- Generative model: For each node:
  - generate all its left children and go recursively into them
  - generate the left STOP sign
  - generate all its right children and go recursively into them
  - generate the right STOP sign



Dependency Model with Valence

- $P_{STOP}(STOP|h,dir,adj)$ ... probability that no more child of the head *h* will be generated in the direction *dir*
- $P_{CHOOSE}(a|h,dir)$ ... probability of children *a* for the head *h* and direction *dir*
- *adj* ... is something generated in the given direction?

$$P(D(h)) = \prod_{dir \in \{l,r\}} \prod_{a \in deps_D(h,dir)} P_{STOP}(\neg STOP|h, dir, adj)$$

$$P_{CHOOSE}(a|h, dir) P(D(a))$$

$$P_{STOP}(STOP|h, dir, adj)$$

Extended Valency Grammar and Lexicalization

- $P_{CHOOSE}(a|h,dir,adj)$ instead of $P_{CHOOSE}(a|h,dir)$
- Lexicalization: uses wordform+tag instead of tag only
- Smoothing

Progress in 2005 – 2011

- Attachment score on English PTB, WSJ23

Random baseline	4.4%
Left chain baseline	21.0%
Right chain baseline	29.4%
DMV (2005)	35.9%
EVG (2009)	42.6%
Lexicalization (2009)	45.4%
Gillenwater (2010)	53.3%
Blunsom and Cohn (2010)	55.7%
Spitkovsky (2011)	58.4%

MY EXPERIMENTS

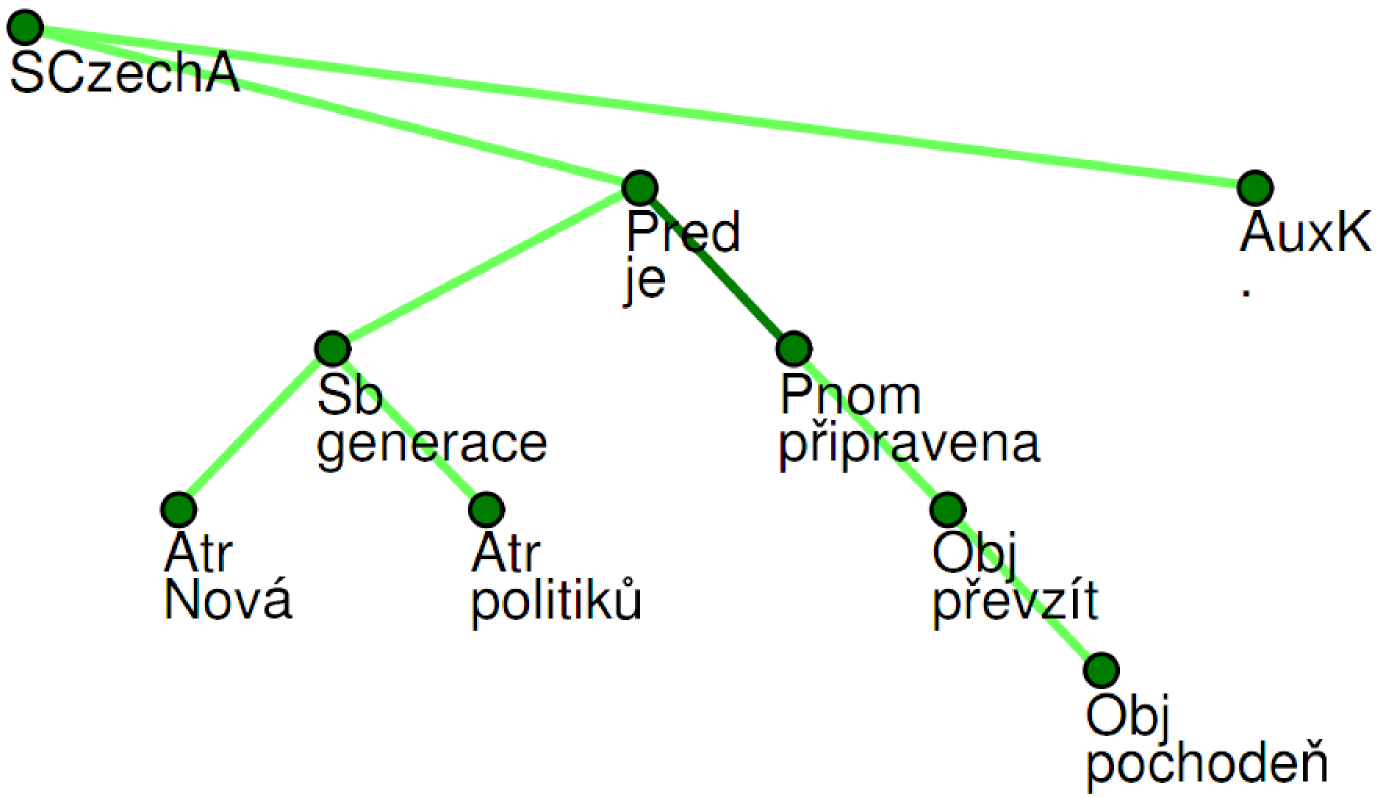
- reducibility feature for recognition of dependent words
- four submodels for modeling dependency trees
- Gibbs sampling algorithm for dependency structure induction

Reducibility feature

- Can we somehow recognize from a text which words are dependents?
- *A word (or a sequence of words) is reducible if the sentence after removing the word(s) remains grammatically correct.*
- Hypothesis: Reducible words (or reducible sequences of words) are leaves (subtrees) in dependency tree.

Reducibility - example

- …



Computing reducibility

- How can we automatically recognize whether a sentence is grammatical or not?
  - Hardly...
- If we have a large corpus, we can search for the needed sentence.
  - it is in the corpus → it is (possibly) grammatical
  - it is not in the corpus → we do not know
- We would like to assign some reducibility scores to the PoS tags (sequences of PoS tags)
  - adjectives and adverbs – high reducibility
  - nouns – middle reducibility
  - verbs – low reducibility

Computing reducibility

- for PoS sequence *g* = [t1, ..., tn]
  - We go through the corpus and search for all its occurrences
  - For each such occurrence, we remove the respective words from the sentence and check in the corpus whether the rest of the sentence occurs at least ones elsewhere in the corpus. If so, then such sequence of words is reducible.
  - r(g) ... number of reducible sequences *g* in the corpus
  - c(g) ... number of all sequences *g* in the corpus

$$R(g) = \frac{1}{N}\frac{r(g)+\sigma_1}{c(g)+\sigma_2}, \quad N = \frac{\sum_{g\in G}(r(g)+\sigma_1)}{\sum_{g\in G}(c(g)+\sigma_2)}$$

$$\sigma_1 = \frac{\sum_{g\in G} r(g)}{\sum_{g\in G} c(g)}, \qquad \sigma_2 = 1$$

Examples of reducibility scores

- Reducibility of Czech PoS tags (1st and 2nd position of PDT tag)

unigrams	R	bigrams	R	trigrams	R
P4	0.00	RR AA	0.00	RR NN Z:	0.00
RV	0.00	Z: J,	0.00	NN RR AA	0.00
Vp	0.06	Vp NN	0.00	NN AA NN	0.16
Vf	0.06	VB NN	0.12	AA NN RR	0.23
P7	0.16	NN Vp	0.13	NN RR NN	0.46
J,	0.24	NN VB	0.18	NN J^ NN	0.46
RR	0.28	NN RR	0.22	AA NN NN	0.47
VB	0.33	NN AA	0.23	NN Z: Z:	0.48
NN	0.72	NN J^	0.62	NN Z: NN	0.52
J^	1.72	AA NN	0.62	NN NN NN	0.70
C=	1.85	NN NN	0.70	AA AA NN	0.72
PD	2.06	NN Z:	0.97	AA NN Z:	0.86
AA	2.22	Z: NN	1.72	NN NN Z:	1.38
Dg	3.21	Z: Z:	1.97	RR NN NN	2.26
Z:	4.01	J^ NN	2.05	RR AA NN	2.65
Db	4.62	RR NN	2.20	Z: NN Z:	8.32

Examples of reducibility scores

- Reducibility of English PoS tags

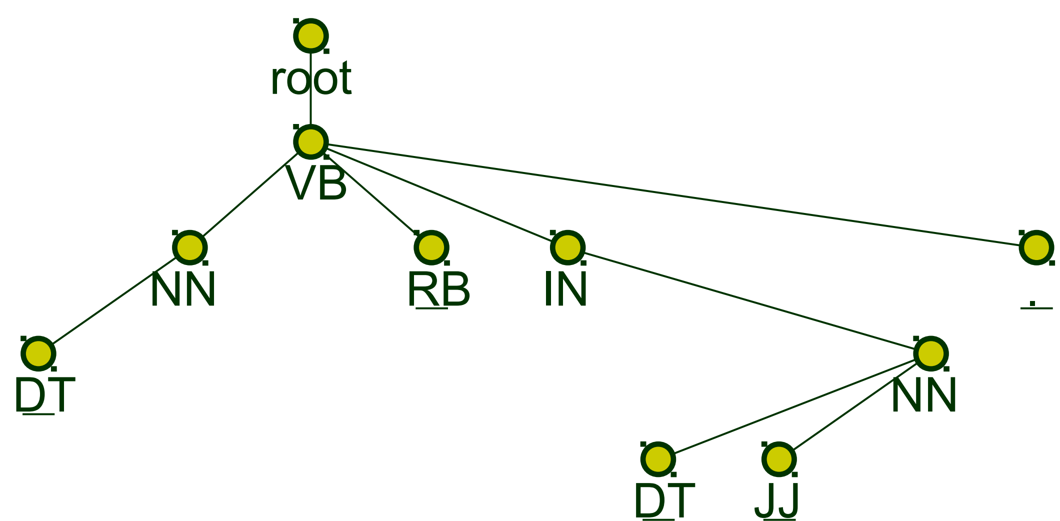
unigrams	R	bigrams	R	trigrams	R
VB	0.04	VBN IN	0.00	IN DT JJ	0.00
TO	0.07	IN DT	0.02	JJ NN IN	0.00
IN	0.11	NN IN	0.04	NN IN NNP	0.00
VBD	0.12	NNS IN	0.05	VBN IN DT	0.00
CC	0.13	JJ NNS	0.07	JJ NN .	0.00
VBZ	0.16	NN .	0.08	DT JJ NN	0.04
NN	0.22	DT NNP	0.09	DT NNP NNP	0.05
VBN	0.24	DT NN	0.09	NNS IN DT	0.14
.	0.32	NN ,	0.11	NNP NNP .	0.15
NNS	0.38	DT JJ	0.13	NN IN DT	0.23
DT	0.43	JJ NN	0.14	NNP NNP ,	0.46
NNP	0.78	NNP .	0.15	IN DT NNP	0.55
JJ	0.84	NN NN	0.22	DT NN IN	0.59
RB	2.07	IN NN	0.67	NNP NNP NNP	0.64
,	3.77	NNP NNP	0.76	IN DT NN	0.80
CD	55.6	IN NNP	1.81	IN NNP NNP	4.27

Dependency tree model

- Consists of four submodels
  - edge model, fertility model, distance model, subtree model
- Simplification
  - we use only PoS tags, we don't use word forms
  - we induce projective trees only

FERTILITY:
P(fert|tagH)

EDGE:
P(tagD|tagH)



Edge model

- P(dependent tag | direction, parent tag)
  - Chinese restaurant process
  - If an edge has been frequent for in the past, it is more likely to be generated again
  - Dirichlet hyperparameter β

$$P_e(t_j|t_i, d_j) = \frac{c^{-i}(\text{``}t_i, t_j, d_j\text{''}) + \beta}{c^{-i}(\text{``}t_i, d_j\text{''}) + \beta|T|},$$

Fertility model

- P(number of children | parent tag)
  - Chinese restaurant process
  - Hyperparameter α_e is divided by a frequency of a word form

$$P'_f(f_i|t_i, w_i) = \frac{c^{-i}(\text{``}t_i, f_i\text{''}) + \frac{\alpha_e}{F(w_i)} P_0(f_i)}{c^{-i}(\text{``}t_i\text{''}) + \frac{\alpha_e}{F(w_i)}},$$

Distance model

- Longer edges are less probable.

$$P_d(i,j) = \frac{1}{\epsilon_d}\left(\frac{1}{|i-j|}\right)^{\gamma}$$

Subtree model

- The higher reducibility score the subtree (or leaf) has, the more probable it is.

$$P_s(i) = \frac{1}{\epsilon_s} R(desc(i))^{\delta}$$

Probability of treebank

- The probability of the whole treebank, which we want to maximize
  - Multiplication over all nodes and models

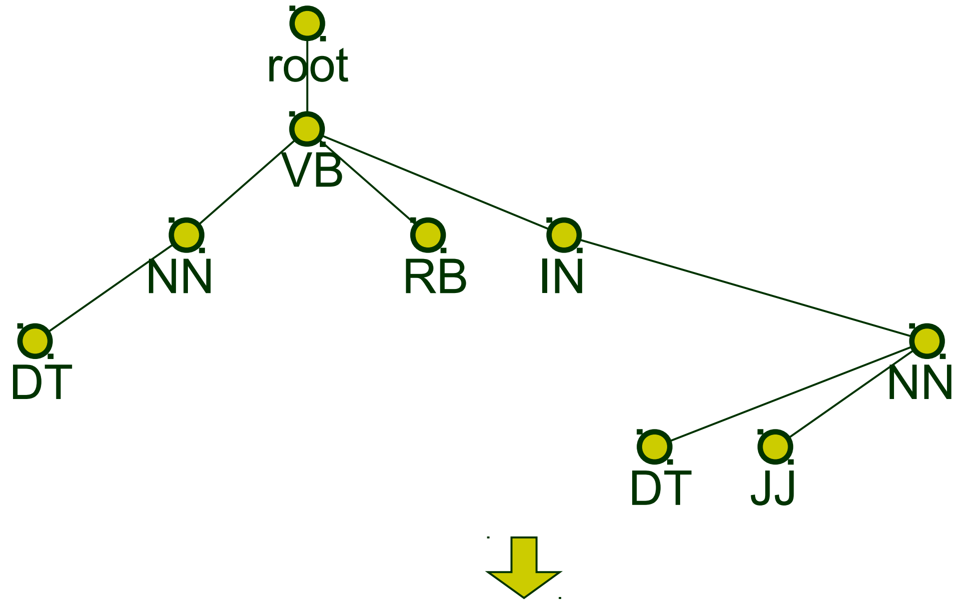
$$P_{treebank} = \prod_{i=1}^{n} (P'_f(f_i|t_i, w_i)$$
$$P_e(t_i|t_{\pi(i)}, d_i)$$
$$P_d(i, \pi(i))$$
$$P_s(i)),$$

Gibbs sampling

- Iterative approximation algorithm which helps with searching for the most probable solution
  - Often used in unsupervised machine learning
- First, dependency trees for all the sentences in the corpus are initialized randomly.
  - We can compute the initial probability of the treebank
- We are doing a small changes in the treebank
  - We pick a node and randomly change the dependency structure of its neighbourhood by weighted coin flip
  - The changes that lead to higher treebank probability are more probable than the changes that lead to lower probability
- After more than 200 iterations (200 small changes for the each node), the dependency trees converge

Gibbs sampling – bracketing notation

- Each projective dependency tree can be expressed by a unique bracketing.
  - Each bracket pair belongs to one node and delimits its descendants from the rest of the sentence.
  - Each bracketed segment contains just one word that is not embedded deeper; this node is the segment head.



(((DT) NN) VB (RB) (IN ((DT) (JJ) NN)))

Gibbs sampling – small change

- Choose one non-root node and remove its bracket
- Add another bracket which does not violate projective tree constraints

(VB)	0.0009
(VB (RB))	0.0011
(((DT) NN) VB)	0.0016
(((DT) NN) VB (RB))	0.0018
(IN)	0.0006
(IN ((DT) (JJ) NN))	0.0023
((RB) IN ((DT) (JJ) NN))	0.0012
((RB) IN)	0.0004

((((DT) NN) VB (RB)) IN ((DT) (JJ) NN))

Gibbs sampling

- After 100-200 iterations, the trees converge.
  - we can pick the actual treebank as it is after the last iteration
  - we can average the last (100) iterations using maximum spanning tree algorithm

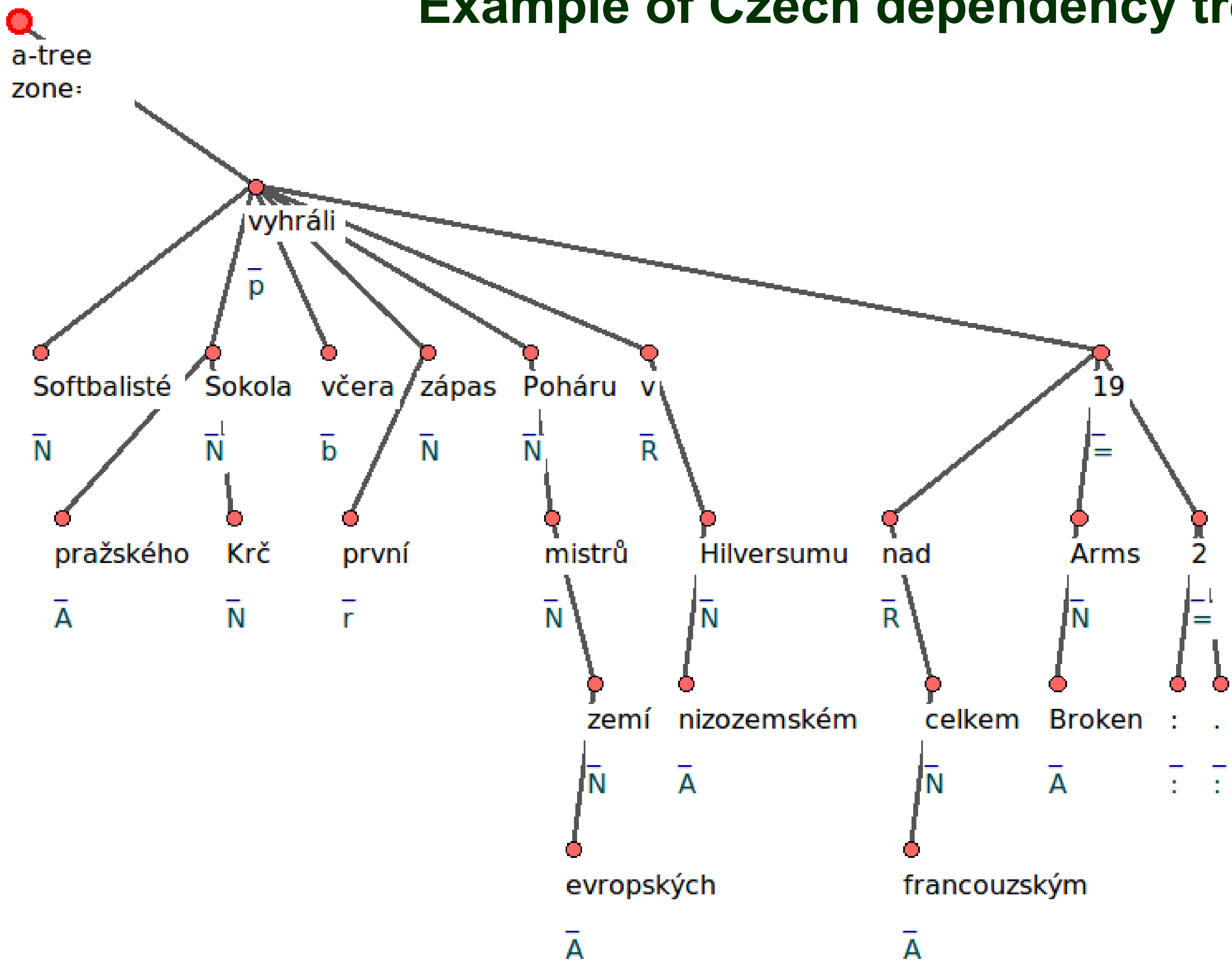
Evaluation and Results

- Directed attachment scores on CoNLL 2006/2007 test data
  - comparison with Spitkovsky 2011 (possibly state-of-the-art)

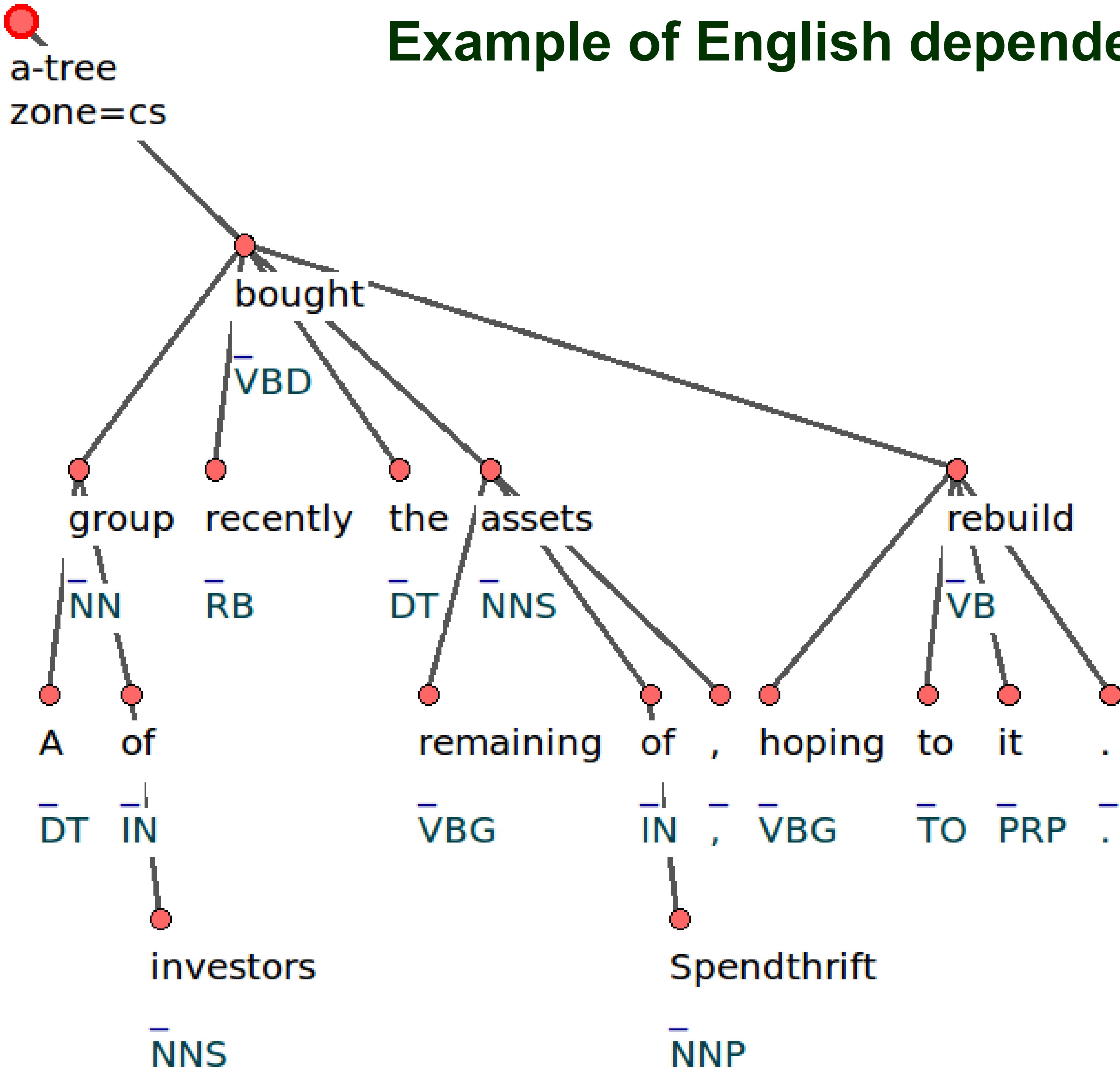
language	*spi11*	*our*
Arabic	16.6	**26.5**
Basque	24.0	**26.8**
Bulgarian	43.9	**46.0**
Catalan	**59.8**	47.0
Czech	27.7	**49.5**
Danish	38.3	**38.6**
Dutch	27.8	**44.2**
English	45.2	**49.2**
German	30.4	**44.8**

language	*spi11*	*our*
Greek	13.2	**20.2**
Hungarian	34.7	**51.8**
Italian	**52.3**	43.3
Japanese	50.2	**50.8**
Portuguese	36.7	**50.6**
Slovenian	**32.2**	18.0
Spanish	50.6	**51.9**
Swedish	**50.0**	48.2
Turkish	**35.9**	15.7

Example of Czech dependency tree



Example of English dependency tree



Conclusions

- We have an unsupervised dependency parser, which has been tested on 18 different languages.
- We achieved higher attachment scores for 13 of them.
  - Compared with previous results reported by Spitkovsky (2011)

Thank you for your attention.