Extracting Dependency Relations for Opinion Mining

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa

Semantic Document Analysis

- **Question Answering**
  - **Return precise answer to natural language queries**
- **Relation Extraction**
- **Intent Mining**
  - **assess the attitude of the document author with respect to a given subject, e.g. *problem* (*description*, *solution*), *agreement* (*assent*, *dissent*), *preference* (*likes*, *dislikes*), *statement* (*claim*, *denial*)**
  - **Opinion mining: attitude is a positive or negative opinion**

Is NLP needed?

- **Many Information Retrieval tasks can do without**
  - **Document retrieval, categorization, summarization, information filtering**
- **Focus on Document Retrieval reduces the need for NLP techniques**
  - **Discourse factors can be ignored**
  - **Redundant words perform word-sense disambiguation**
- **Lack of robustness:**
  - **NLP techniques are typically not as robust as word indexing**

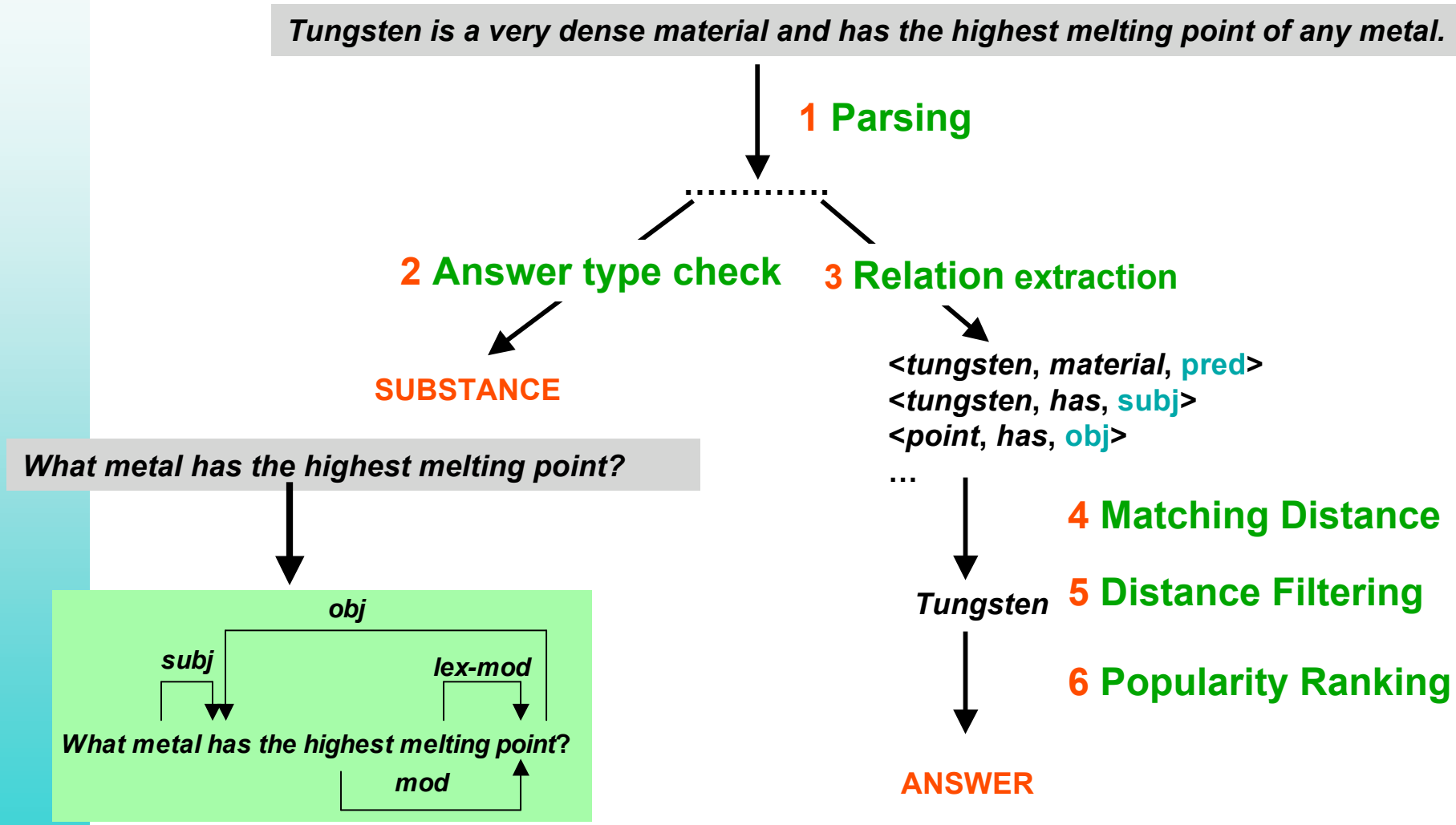
Question Answering is Different

- **Search Engines return list of (possibly) *relevant* documents**
- **Users still to have to dig through returned list to find answer**
- **QA: give the user a (short) answer to their question, perhaps supported by evidence**

Best QA Systems at TREC 2005

- **LCC (70% accuracy), Singapore University (66%)**
- **Both perform parsing of question and candidate answers**
- **Singapore analyzes similarity in dependency trees (similar to PIQASSO from Pisa)**

PiQASso Answer Matching



Parser Requirements

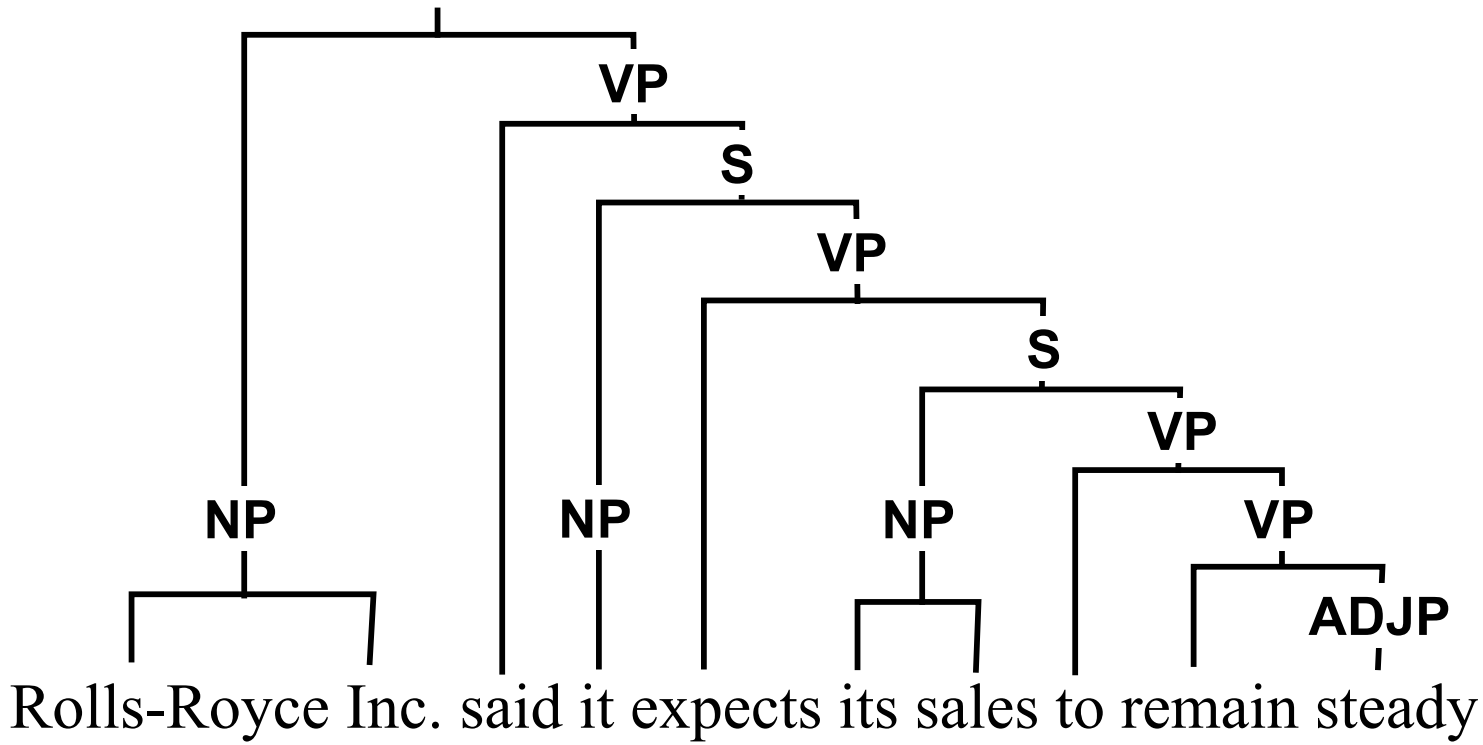
- **Multilanguage**
  - Trainable on annotated corpora
- **Accurate**
  - Close to state of the art
- **Flexible**
  - Retrainable through unannotated corpora
- **Efficient**
  - Hundreds sentence/sec
  - Deterministic bottom-up parser

Statistical Parsers

- **Major breakthrough in computational linguistics in recent years**
- **Parser types:**
  - **Constituent parser**
  - **Dependency parser**

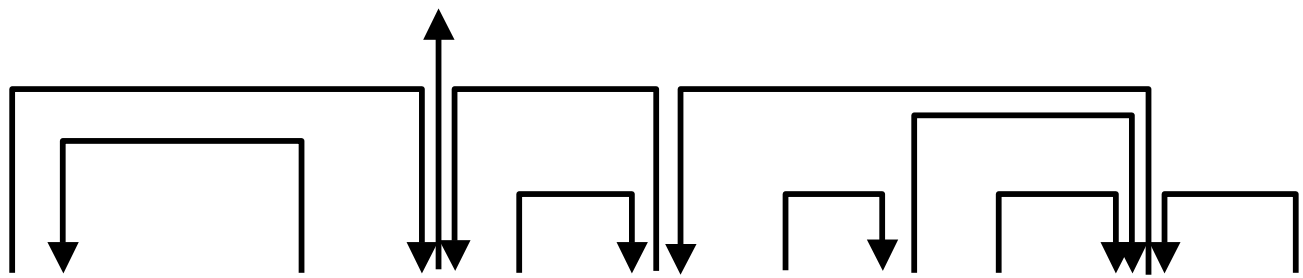
Constituent Parsing

- **Requires Phrase Structure Grammar**
  - **CFG, PCFG, Unification Grammar**
- **Produces phrase structure parse tree**



Dependency Parsing

- **Produces dependency trees**
- **Word-word dependency relations**
- **Far easier to understand and to annotate**



Rolls-Royce Inc. said it expects its sales to remain steady

Linear Parsing Model

- **Three components:**

  GEN **is a function from a string to a set of candidates**

  F maps a candidate to a feature vector

  W **is a parameter vector**

Global Linear Parsing Model

- X: set of sentences
- Y: set of possible parse trees
- Learn function F: $X \rightarrow Y$
- Choose the highest scoring tree as the most plausible:

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}} \Phi(y) \cdot W$$

- Involves just learning weights W

Feature Vector

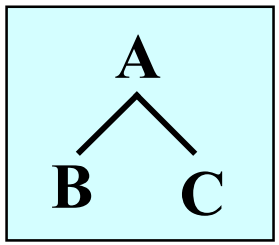
- **A high-dimensional vector of features**
- **Defined through a set of functions $h_1 \ldots h_d$**

$$\mathsf{F}(x) = \langle h_1(x), h_2(x) \ldots h_d(x) \rangle$$

Constituent Parsing

- *GEN*: e.g. CFG (Context-Free Grammar)
- $h_i(x)$ are based on aspects of the tree

e.g.

$h(x)$ = # of times 

occurs in x

Constituent Parsers

- **Probabilistic Generative Model of Language which include parse structure (e.g. Collins 1997)**
- **Conditional parsing models (Charniak 2000; McDonald 2005)**

Generative Dependency Parsing

- *GEN* generates all possible Maximum Spanning Trees
- Select MST with best feature vector
- First order factorization:

  $F(y) = \langle h(0, 1), \ldots h(n\text{-}1, n) \rangle$
- Second order factorization (McDonald 2006):

  $F(y) = \langle h(0, 1, 2), \ldots h(n\text{-}2, n, n) \rangle$

Shift/Reduce Dependency Parser

- **Traditional statistical parsers are trained directly on the task of selecting a parse tree for a sentence**
- **Instead a Shift/Reduce parser is trained and learns the sequence of parse actions required to build the parse tree**

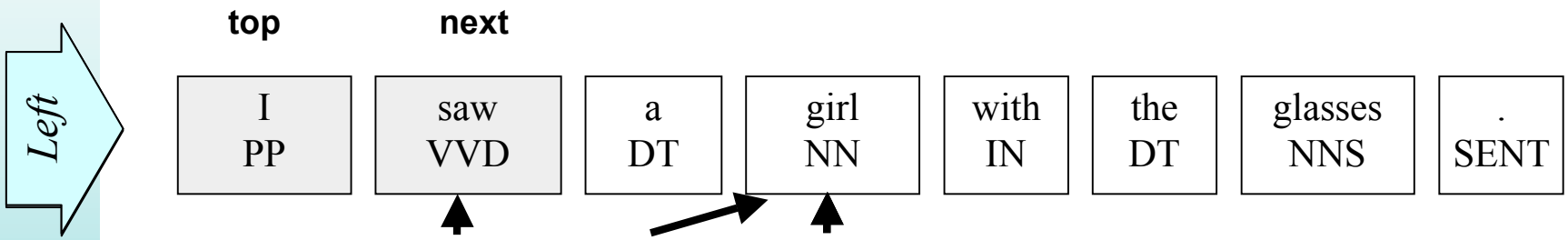
Grammar Not Required

- **A traditional parser requires a grammar for generating candidate trees**
- **A Shift/Reduce parser needs no such grammar**

Parsing as Classification

- **Parsing based on Shift/Reduce actions**
- **Learns from annotated corpus which action to perform at each step**
- **Proposed by (Yamada-Matsumoto 2003) and (Nivre 2003)**
- **Uses only local information, but it can exploit history**

Parser Actions



Dependency Graph

Let $R = \{r_1, \ldots, r_m\}$ be the set of permissible dependency types

A dependency graph for a sequence of words

$W = w_1 \ldots w_n$ is a labeled directed graph

$D = (W, A)$, where

(a) W is the set of nodes, i.e. word tokens in the input string,

(b) A is a set of labeled arcs (w_i, r, w_j), $w_i, w_j \in W$, $r \in R$,

(c) $\forall\ w_j \in W$, there is at most one arc $(w_i, r, w_j) \in A$.

Parser State

The parser state is a quadruple $\langle S, I, T, A \rangle$, where

- S **is a stack of partially processed tokens**
- I **is a list of (remaining) input tokens**
- T **is a stack of temporary tokens**
- A **is the arc relation for the dependency graph**

$(w, r, h) \in A$ **represents an arc** $w \rightarrow h$**, tagged with dependency** r

Parser Actions

Shift

$$\frac{\langle S, n|I, T, A\rangle}{\langle n|S, I, T, A\rangle}$$

Right

$$\frac{\langle s|S, n|I, T, A\rangle}{\langle S, n|I, T, A\cup\{(s, r, n)\}\rangle}$$

Left

$$\frac{\langle s|S, n|I, T, A\rangle}{\langle S, s|I, T, A\cup\{(n, r, s)\}\rangle}$$

Parser Algorithm

- **The parsing algorithm is fully deterministic:**

  Input Sentence: $(w_1, p_1), (w_2, p_2), \ldots, (w_n, p_n)$

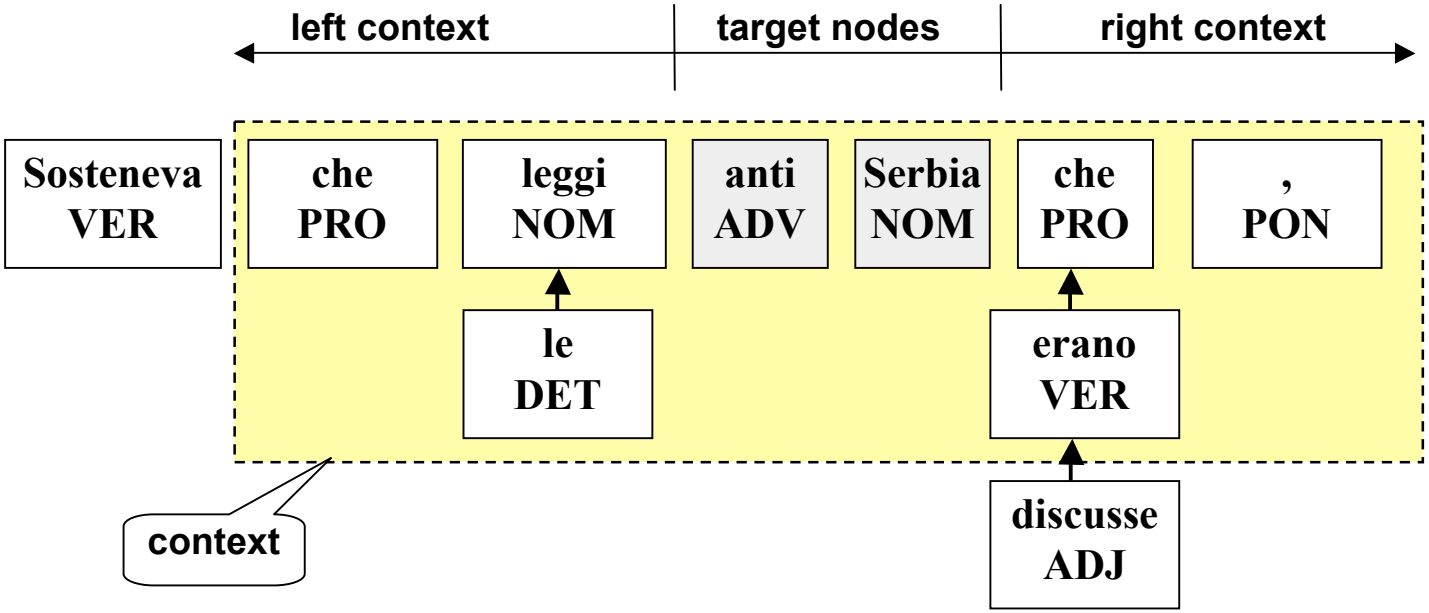
```
S = <>
I = <(w1, p1), (w2, p2), … , (wn, pn)>
T = <>
A = { }
while I ≠ <> do begin
  x = getContext(S, I, T, A);
  y = estimateAction(model, x);
  performAction(y, S, I, T, A);
end
```

Learning Phase

Learning Features

feature	Value
W	word
L	lemma
P	part of speech (POS) tag
M	morphology: e.g. singular/plural
W<	word of the leftmost child node
L<	lemma of the leftmost child node
P<	POS tag of the leftmost child node, if present
M<	whether the rightmost child node is singular/plural
W>	word of the rightmost child node
L>	lemma of the rightmost child node
P>	POS tag of the rightmost child node, if present
M>	whether the rightmost child node is singular/plural

Learning Event



(-3, W, che), (-3, P, PRO),
(-2, W, leggi), (-2, P, NOM), (-2, M, P), (-2, W<, le), (-2, P<, DET), (-2, M<, P),
(-1, W, anti), (-1, P, ADV),
(0, W, Serbia), (0, P, NOM), (0, M, S),
(+1, W, che), (+1, P, PRO), (+1, W>, erano), (+1, P>, VER), (+1, M>, P),
(+2, W, ,), (+2, P, PON)

DeSR Parser Architecture

- **Modular learners architecture:**
  - **MaxEntropy, MBL, SVM, Winnow, Perceptron**
- **Classifier combinations: e.g. multiple MEs, SVM + ME**
- **Features can be selected**

Feature used in Experiments

LemmaFeatures	**-2 -1 0 1 2 3**
PosFeatures	**-2 -1 0 1 2 3**
MorphoFeatures	**-1 0 1 2**
PosLeftChildren	**2**
PosLeftChild	**-1 0**
DepLeftChild	**-1 0**
PosRightChildren	**2**
PosRightChild	**-1 0**
DepRightChild	**-1**
PastActions	**1**

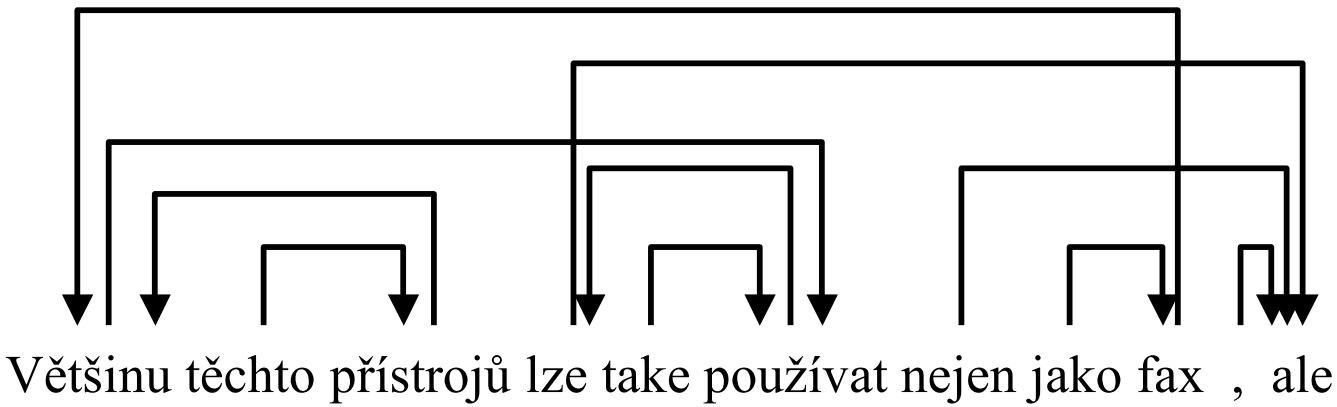
Projectivity

- An arc $w_i \rightarrow w_k$ is projective iff $\forall j, i < j < k$ or $i > j > k$,

  $$w_i \rightarrow^* w_k$$

- A dependency tree is projective iff every arc is projective
- Intuitively: arcs can be drawn on a plane without intersections

Non Projective



Actions for non-projective arcs

Right2

$$\frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle s_1|S, n|I, T, A\cup\{(s_2, r, n)\}\rangle}$$

Left2

$$\frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle s_2|S, s_1|I, T, A\cup\{(n, r, s_2)\}\rangle}$$

Right3

$$\frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle s_1|s_2|S, n|I, T, A\cup\{(s_3, r, n)\}\rangle}$$

Left3

$$\frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle s_2|s_3|S, s_1|I, T, A\cup\{(n, r, s_3)\}\rangle}$$

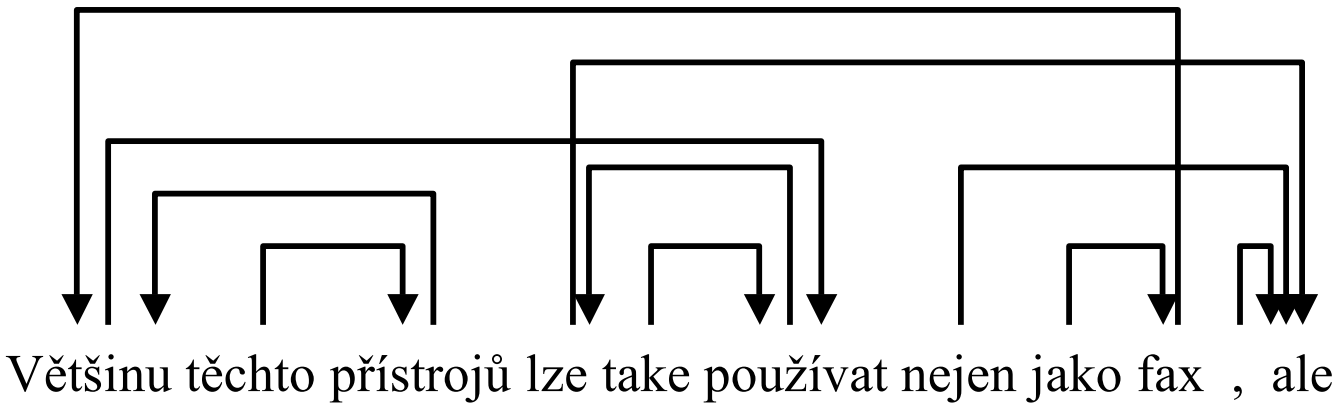
Extract

$$\frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle n|s_1|S, I, s_2|T, A\rangle}$$

Insert

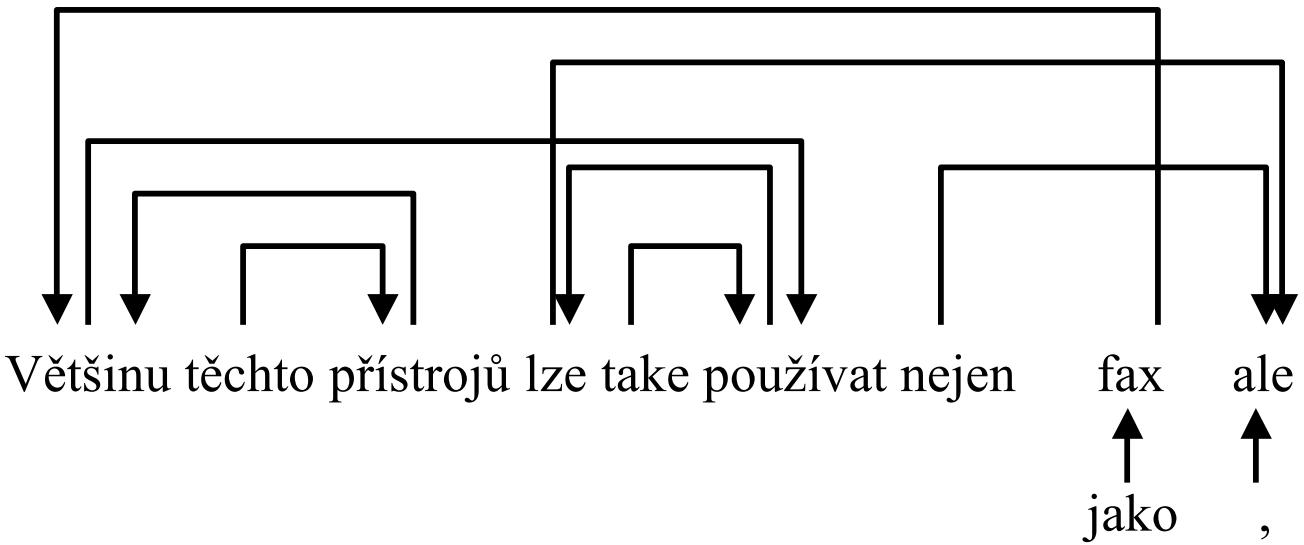
$$\frac{\langle S, I, s_1|T, A\rangle}{\langle s_1|S, I, T, A\rangle}$$

Example

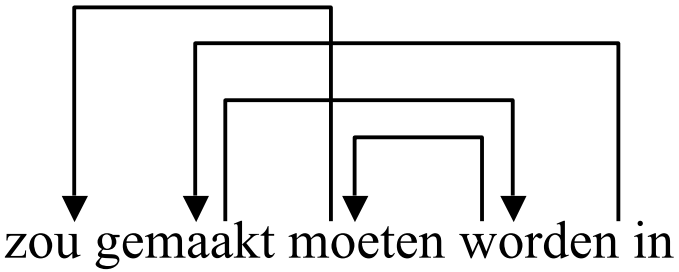


- ***Right2*** **(*nejen* → *ale*) and *Left3* (*fax* → *Většinu*)**

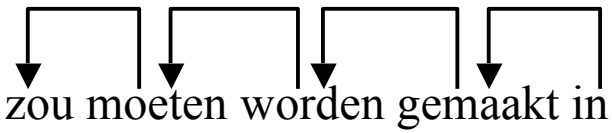
Example


Většinu těchto přístrojů lze take používat nejen fax ale
jako
,

Examples



Extract followed by *Insert*



Effectiveness for Non-Projectivity

- **Training data for Czech contains 28081 non-projective relations**
- **26346 (93%) can be handled by Left2/Right2**
- **1683 (6%) by Left3/Right3**
- **52 (0.2%) require Extract/Insert**

CoNLL-X Shared Task

- **To assign labeled dependency structures for a range of languages by means of a fully automatic dependency parser**
- **Input: tokenized and tagged sentences**
- **Tags: token, lemma, POS, morpho features, ref. to head, dependency label**
- **For each token, the parser must output its head and the corresponding dependency relation**

CoNLL-X: Collections

	Ar	*Cn*	*Cz*	*Dk*	*Du*	*De*	*Jp*	*Pt*	*Sl*	*Sp*	*Se*	*Tr*	*Bu*
K tokens	**54**	**337**	**1,249**	**94**	**195**	**700**	**151**	**207**	**29**	**89**	**191**	**58**	**190**
K sents	**1.5**	**57.0**	**72.7**	**5.2**	**13.3**	**39.2**	**17.0**	**9.1**	**1.5**	**3.3**	**11.0**	**5.0**	**12.8**
Tokens/sentence	**37.2**	**5.9**	**17.2**	**18.2**	**14.6**	**17.8**	**8.9**	**22.8**	**18.7**	**27.0**	**17.3**	**11.5**	**14.8**
CPOSTAG	**14**	**22**	**12**	**10**	**13**	**52**	**20**	**15**	**11**	**15**	**37**	**14**	**11**
POSTAG	**19**	**303**	**63**	**24**	**302**	**52**	**77**	**21**	**28**	**38**	**37**	**30**	**53**
FEATS	**19**	**0**	**61**	**47**	**81**	**0**	**4**	**146**	**51**	**33**	**0**	**82**	**50**
DEPREL	**27**	**82**	**78**	**52**	**26**	**46**	**7**	**55**	**25**	**21**	**56**	**25**	**18**
% non-project. relations	**0.4**	**0.0**	**1.9**	**1.0**	**5.4**	**2.3**	**1.1**	**1.3**	**1.9**	**0.1**	**1.0**	**1.5**	**0.4**
% non-project. sentences	**11.2**	**0.0**	**23.2**	**15.6**	**36.4**	**27.8**	**5.3**	**18.9**	**22.2**	**1.7**	**9.8**	**11.6**	**5.4**

CoNLL: Evaluation Metrics

- **Labeled Attachment Score (LAS)**
  - **proportion of "scoring" tokens that are assigned both the correct head and the correct dependency relation label**
- **Unlabeled Attachment Score (UAS)**
  - **proportion of "scoring" tokens that are assigned the correct head**

Shared Task Unofficial Results

Language	Maximum Entropy				MBL			
	LAS %	UAS %	Train sec	Parse sec	LAS %	UAS %	Train sec	Parse sec
Arabic	56.43	**70.96**	181	2.6	59.70	**74.69**	24	950
Bulgarian	82.88	**87.39**	452	1.5	79.17	**85.92**	88	353
Chinese	81.69	**86.76**	1,156	1.8	72.17	**83.08**	540	478
Czech	62.10	**73.44**	13,800	12.8	69.20	**80.22**	496	13,500
Danish	77.49	**83.03**	386	3.2	78.46	**85.21**	52	627
Dutch	70.49	**74.99**	679	3.3	72.47	**77.61**	132	923
Japanese	84.17	**87.15**	129	0.8	85.19	**87.79**	44	97
German	80.01	**83.37**	9,315	4.3	79.79	**84.31**	1,399	3,756
Portuguese	79.40	**87.70**	1,044	4.9	80.97	**87.74**	160	670
Slovene	61.97	**74.78**	98	3.0	62.67	**76.60**	16	547
Spanish	72.35	**76.06**	204	2.4	74.37	**79.70**	54	769
Swedish	78.35	**84.68**	1,424	2.9	74.85	**83.73**	96	1,177
Turkish	58.81	**69.79**	177	2.3	47.58	**65.25**	43	727

CoNLL-X: Comparative Results

	LAS		UAS	
	Average	Ours	Average	Ours
Arabic	**59.94**	59.70	73.48	**74.69**
Bulgarian	79.98	**82.88**	85.89	**87.39**
Chinese	78.32	**81.69**	84.85	**86.76**
Czech	67.17	**69.20**	77.01	**80.22**
Danish	78.31	**78.46**	84.52	**85.21**
Dutch	70.73	**72.47**	75.07	**77.71**
Japanese	**85.86**	85.19	**89.05**	87.79
German	78.58	**80.01**	82.60	**84.31**
Portuguese	80.63	**80.97**	86.46	**87.74**
Slovene	**65.16**	62.67	76.53	**76.60**
Spanish	73.52	**74.37**	77.76	**79.70**
Swedish	76.44	**78.35**	84.21	**84.68**
Turkish	55.95	**58.81**	69.35	**69.79**

Average scores from 36 participant submissions

Performance Comparison

- **Running an SVM-based parser (Maltparser 0.4) on same Xeon 2.8 MHz machine**
- **Training on swedish/talbanken:**
  - **390 min**
- **Test on CoNLL swedish test set (~5000 tokens):**
  - **13 min**

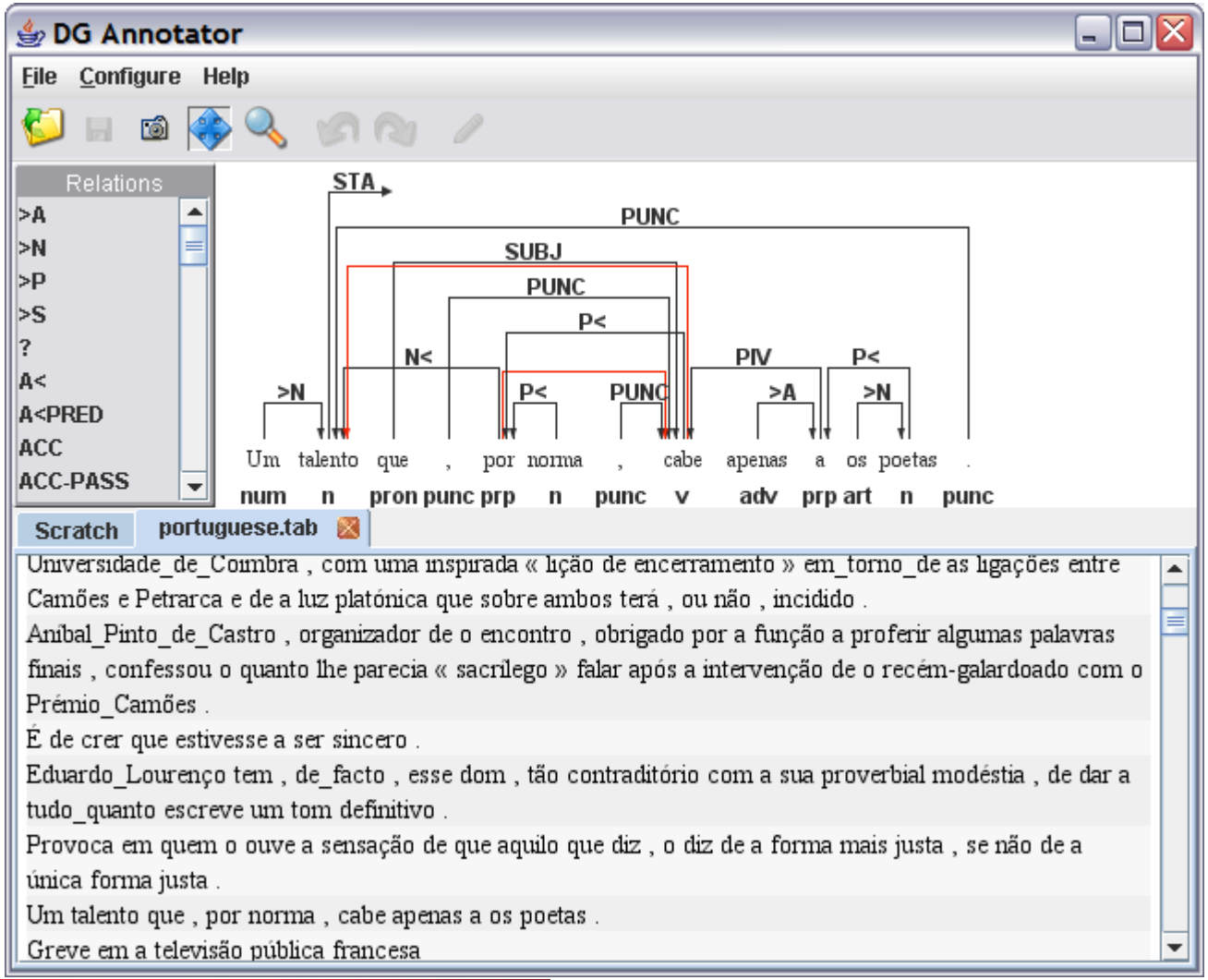
Italian Treebank

- **Official Announcement:**
  - **CNR ILC has agreed to provide the SI-TAL collection for use at CoNLL**
- **Working on completing annotation and converting to CoNLL format**
- **Semiautomated process: heuristics + manual fixup**

DgAnnotator

- **A GUI tool for:**
  - **Annotating texts with dependency relations**
  - **Visualizing and comparing trees**
  - **Generating corpora in XML or CoNLL format**
  - **Exporting DG trees to PNG**
- **Demo**
- **Available at: http://medialab.di.unipi.it/Project/QA/Parser/DgAnnotator/**

DgAnnotator Example



TREC Blog 2006

- **Resources**
- **Collection of blogs**
  - **from a crawl of 100,649 RSS/Atom feeds, over 11 weeks**
  - **Several languages**
  - **Includes spam**
- **Set of 4000 opinionated words, with both positive/negative weights**
  - **Computed by [Esuli & Sebastiani, 2005] using classifier trained on WordNet glosses**

Experiments

- **Baseline: normal IR search**
- **Opinion rank: rank documents on number of opinionated words they contain**
- **Opinion proximity: weight terms according to proximity with opinionated words**

Sample Topics

- **Larry Summers**
- **Macbook pro**
- **Skype 2.0**
- **Colbert Report**

Examples

Baseline Query: skype

- **Title: Skype to release free video phone**

Opinion Rank Query: skype

- **I like my Macbook: I immediately installed Skype.**

Opinionated proximity

Query: proximity 3 [OPINIONATED:* skype]

- **Skype Doesn't Want My Money**
- **… guys working on Skype and no one really took it seriously**
- **Para aquellos que no conozcan Skype**

Using Dependency Relations

- **Classification problem**
- **Training on frequent-subpatterns from sentences as features**
- **Exploit the dependency tree of a sentence to extract frequent sub-patterns**
- **Patterns are obtained by pruning parse trees**

References

- G. Attardi. 2006. Experiments with a Multilanguage Non-projective Dependency Parser. In Proc. CoNLL-X.
- H. Yamada, Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of IWPT-2003*.
- J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-2003*, pages 149–160.
- A. Esuli, F. Sebastiani. 2005. Determining the semantic orientation of terms through gloss analysis. In Proc. of CIKM-05.