



MACHINE LEARNING TECHNIQUES APPLIED TO DEPENDENCY PARSING

Edoardo Maria Ponti
Theoretical and Applied Linguistics
University of Pavia

TABLE OF CONTENTS

- ❖ Dependency vs Constituency
- ❖ Treebanks
- ❖ Languages and Their Complexity
- ❖ Supervised Machine Learning
- ❖ A Case of Study: Medieval Latin
- ❖ Postprocessing Techniques: Revision and Combination
- ❖ New Improvements

WHY PARSING?

Parsing natural language gives structure to a sentence, which in turn allows to access its meaning.

(A few) applications:

- ❖ Data mining: extracting information (and storing in database)
- ❖ Question answering and research engine
- ❖ Automatic translation
- ❖ Grammar checkers in text editors

DEPENDENCY VS CONSTITUENCY

A **constituency** parser breaks a sentence into sub-phrases.

Non-terminal nodes are phrase categories (e.g. Noun Phrase, Verb Phrase, etc.) and the root S. Terminal nodes are words.

Sentence thought of in Aristotelian terms as a subject and a predicate.

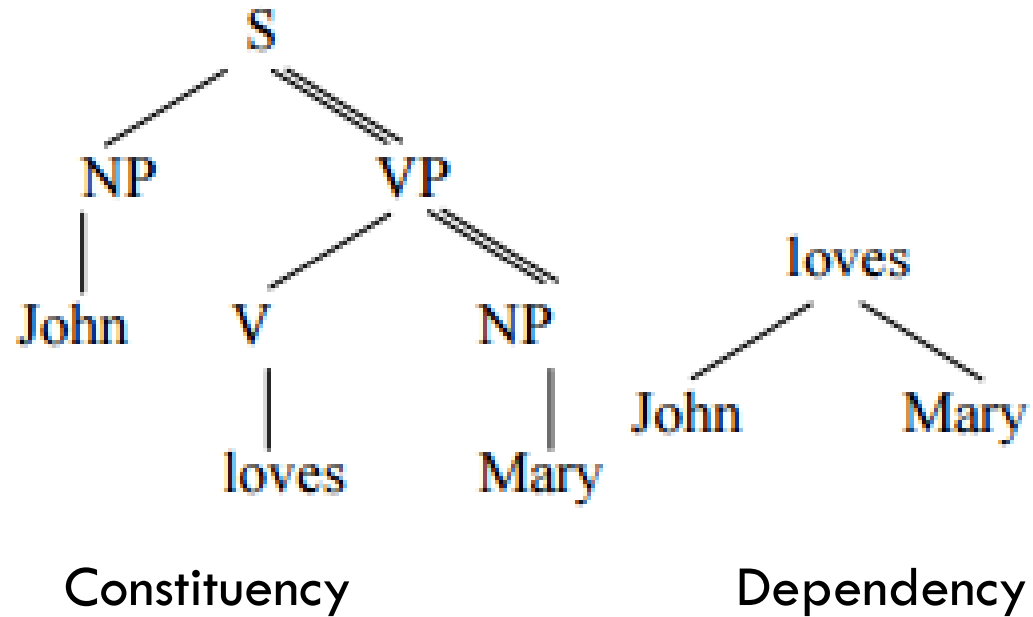
Number of words $>$ number of nodes.

A **dependency** parser links words in a sentence and assign labels to these relations.

Just terminal nodes. Labels mark grammatical functions (e.g. Subject, Predicate, etc.)

Sentence thought of in Fregean terms as a central predicate with its argument.

Number of words $=$ number of nodes.



In last years, researchers tended to orient towards dependency parsing and disjointed this task from natural language generation.

Different theoretical backgrounds underly the different representations.
Chomsky inspired constituency-based grammars, Tesnière dependency-based ones.

Cer (2010)

Dependency parsing is way faster than constituency parsing.

Linear (at most, $O(n^3)$) complexity vs $O(n^5)$ of CKY-style dynamic programming for lexicalised models.

Kahane (2012)

Dependency parsing, contrary to constituency parsing, can deal with non-projectivity (see further) without complex mechanisms such as transformation and movement.

Furthermore, it is more close to the interface with semantics, and obey to valency constraints and account for multi-word expressions.

Finally, it is more grounded on cross-linguistic comparison. Some languages lack constituency: this property is called non-configurationality.

DEPENDENCY TREEBANKS

Dependency = an oriented relation between two words, a head and a dependent, labelled by syntactic function.

Tree = a set of the words and dependency relations of a sentence. The main predicate is the root.

Treebank = a collection of sentences with a syntactic annotation.

Graphic tree format:

Words = nodes

Dependency relations = arrows and labels

Linear order = precedence left-to-right

CoNLL format:

Devised for a shared task on parsing. That of 2007 edition is quite standard:

ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL
----	------	-------	---------	--------	-------	------	--------

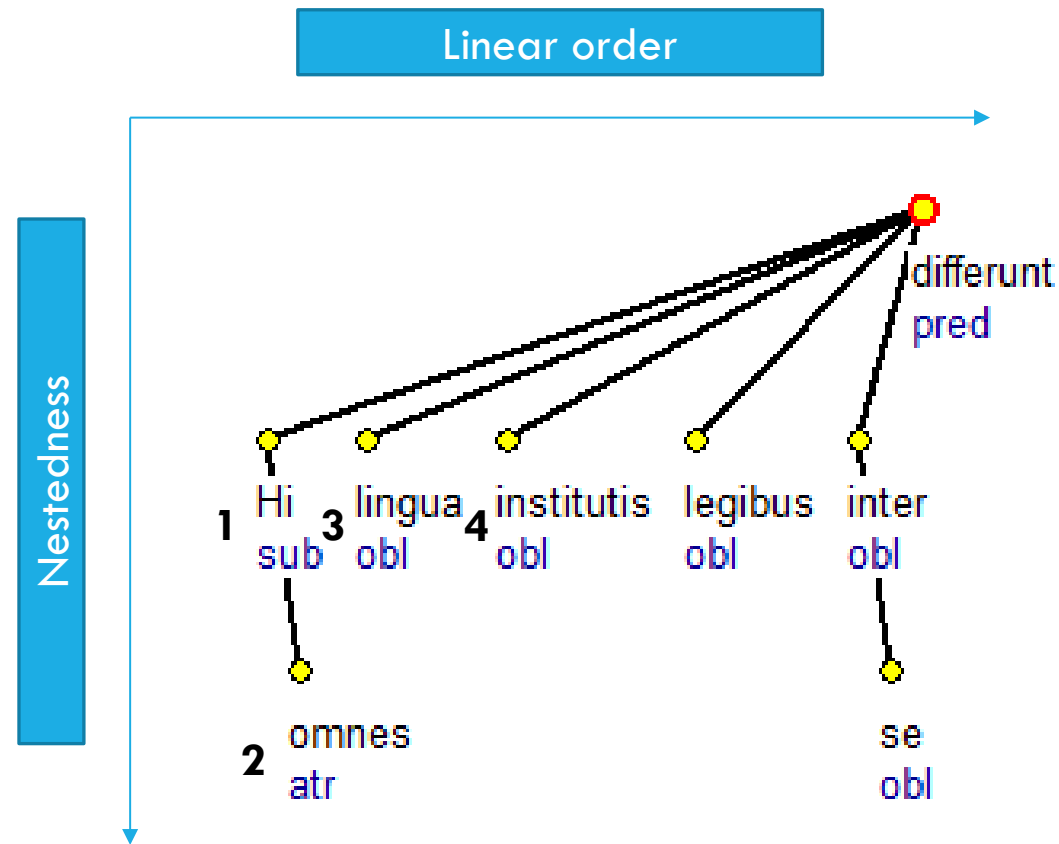
Words = forms, one per row

Word information = values of the attributes above, one per column

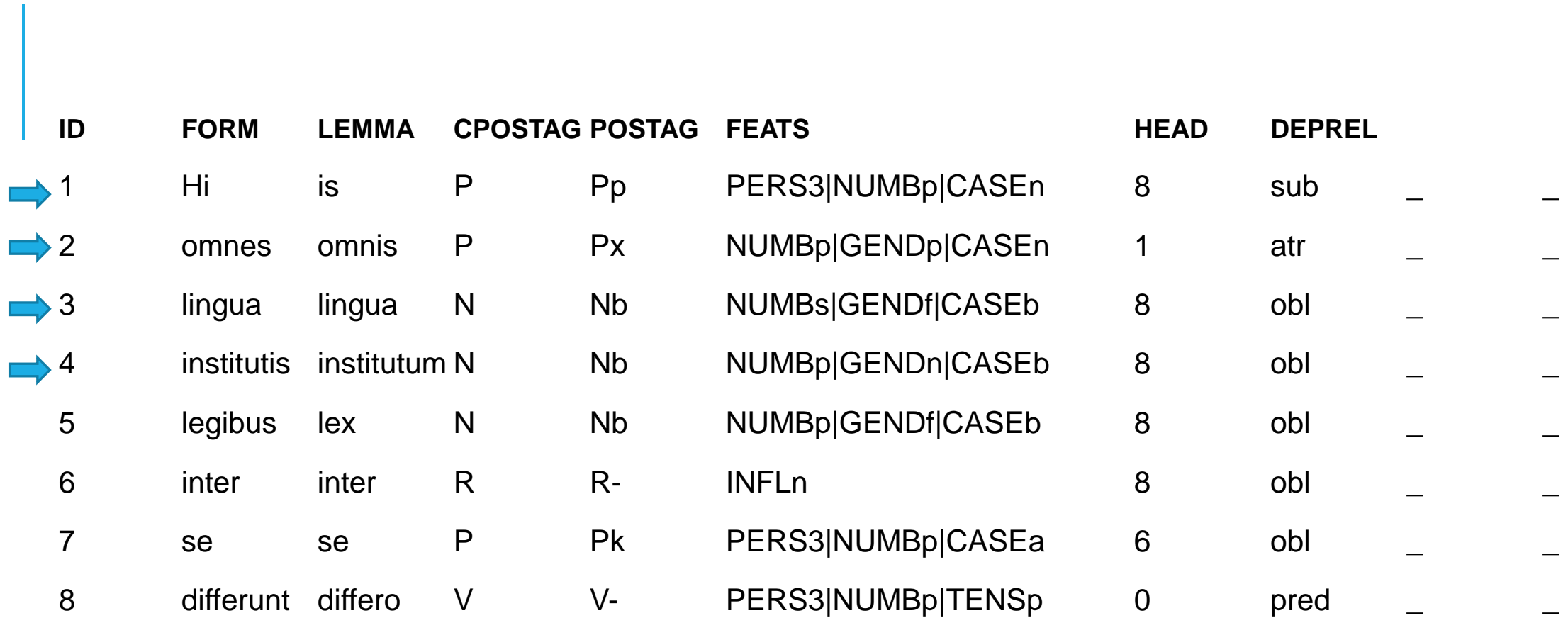
Dependency relations = head and syntactic function (i.e. deprel) columns

Linear order = id column

Julius Caesar's *De Bello Gallico* (graphic tree format)



Julius Caesar's *De Bello Gallico* (CoNLL format)



ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL		
1	Hi	is	P	Pp	PERS3 NUMBp CASEn	8	sub	—	—
2	omnes	omnis	P	Px	NUMBp GENDp CASEn	1	atr	—	—
3	lingua	lingua	N	Nb	NUMBs GENDf CASEb	8	obl	—	—
4	institutis	institutum	N	Nb	NUMBp GENDn CASEb	8	obl	—	—
5	legibus	lex	N	Nb	NUMBp GENDf CASEb	8	obl	—	—
6	inter	inter	R	R-	INFLn	8	obl	—	—
7	se	se	P	Pk	PERS3 NUMBp CASEa	6	obl	—	—
8	differunt	differo	V	V-	PERS3 NUMBp TENS	0	pred	—	—

Different standard annotation styles.

Annotation style = 1) rules for head selection and 2) set of syntactic functions.

Main ones = **Prague** (PRG) and **Stanford** (USD).

Differences:

1) Set member equivalence from Rosa et al. (2014)

PRG	USD
Pred	root
Sb	nsubj, csubj, nsubjpass, csubjpass
Obj, Pnom, Atv, AtvV, AuxR	obj, ccomp, xcomp
Adv, AuxO, AuxY, AuxZ	advmod, nmod, advcl, nfincl, mwe
Atr	amod, nmod, nummod, relcl, nfincl

2) Handling of specific constructions:

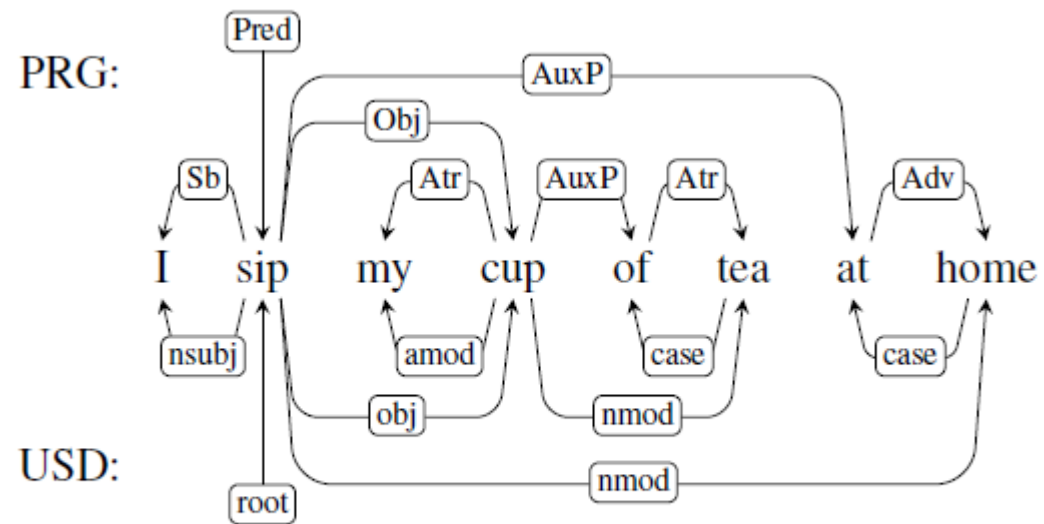


Figure 1: Labeling of prepositional phrases.

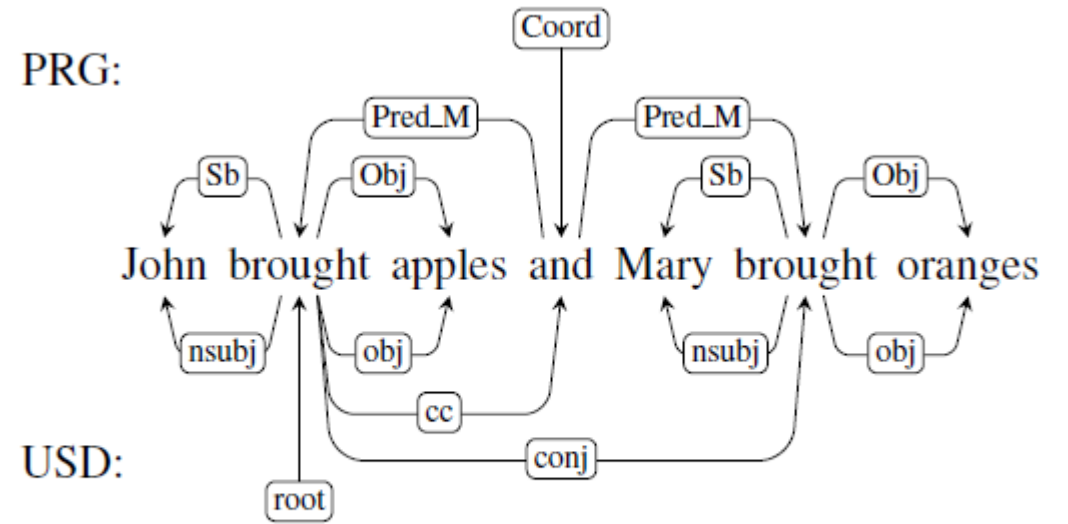


Figure 2: Coordination without ellipsis.

The coordinating conjunction (here 'and') is explicit. Otherwise, if implicit, members are coordinated with ellipsis.

ANCIENT INDO-EUROPEAN LANGUAGES AND THEIR COMPLEXITY

Trebanks by individual language

in (quasi-)Prague annotation style:

Thomas Aquinas' Medieval Latin = Index Thomisticus Treebank (IT-TB)

Classical and Late Latin = Latin Dependency Treebank (LDT)

Ancient Greek = Ancient Greek Treebank (AGT)

Armenian, Gothic, Old Church Slavonic = Pragmatic Resources in Old Indo-European Languages Treebank (PROIEL)

According to Mambrini & Passarotti (2013),
ancient indo-european languages have:

- ❖ free (better, pragmatically constrained) word order.
- ❖ discontinuous constituents.
- ❖ rich fusive morphology. A single morpheme often encodes more than one meaning. E.g. in *Hēródotos*, -os means: singular number, nominative case, masculine gender.

This results in a high degree of non-projectivity.

Non-projectivity = (informal) Presence of crossing arc pairs in a sentence.

Rate in Ancient Greek > Latin > modern languages.

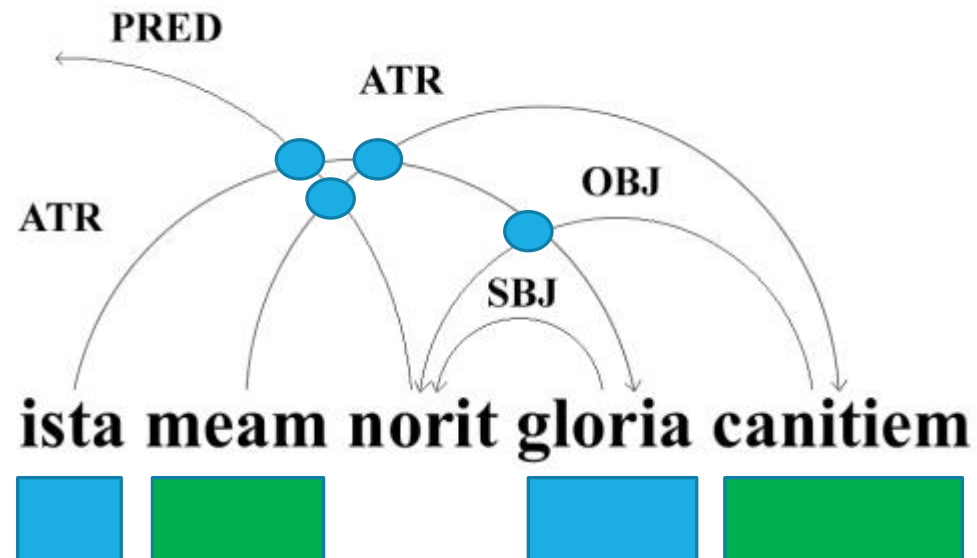
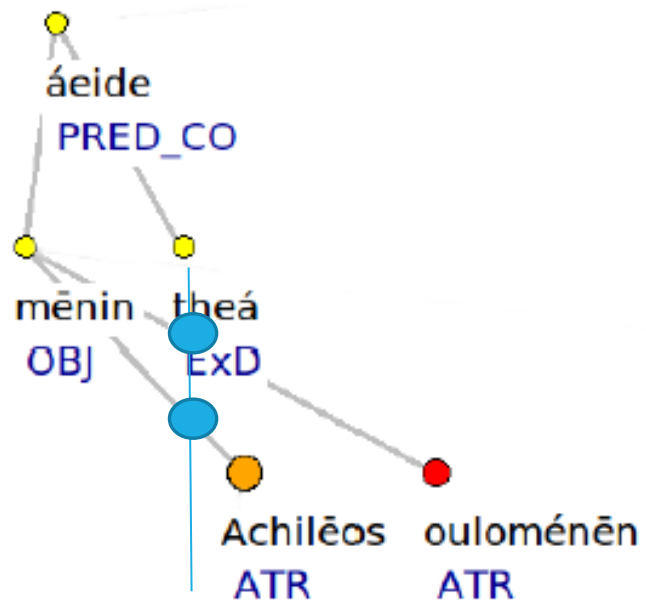
Graphically non-projectivity if present can be visualized in two ways:

1) Imagine to project all nodes down to the lowest layer of nestedness.

At least one of them crosses at least one dependency arc.

2) Compact a tree to a single layer. Dependency arcs cross.

From *Mambrini & Passarotti (2013)*:



$i \rightarrow j = j$ (child) depends on i (parent)

(i, j) = linear array of nodes from i to j

Descendants = set of nodes reachable through n arcs from i

$Subtree_i$ = restriction of T (tree) to the descendants of i

Projectivity = $i \rightarrow j \wedge v \in (i, j) \Rightarrow v \in Subtree_i$

Each dependency subtree should cover a linearly contiguous region of the sentence.

If v violates this condition, v is 'in a gap'.

Non-contiguous regions are highlighted in the previous examples:  and 

Language	Total edges	# Non-proj. edges	% Non-proj edges
Ancient Greek	301848	45731	15,15
Czech	1105437	23570	2,13

Language	Trees	gd0	gd1	gd2	gd3	gd 4	ed0	ed1	ed2	ed3	ed4
Ancient Greek	24825	25,2	68,3	6,1	7	0,3	25,2	43,7	14,2	7,1	3,9
Czech	73088	76,9	22,7	0,4	0	0	76.85	22,7	0,4	0,1	0

Edge-degree = number of nodes in a gap for a given non-projective arc.

Gap-degree = number of intervals in a block.

Block = longest non-empty sequence of nodes chained by dependencies.

Interval = distance between a head and a dependent is linearly more than 1.

E.g. the sequence *norit-canitiem-meam* is interrupted twice, by *gloria* and *norit*.

Hence gap-degree = 2

The dependency *norit-canitiem* is interrupted by *gloria*. Hence edge-degree = 1.

SUPERVISED MACHINE LEARNING

Hladká & Holub (2015)

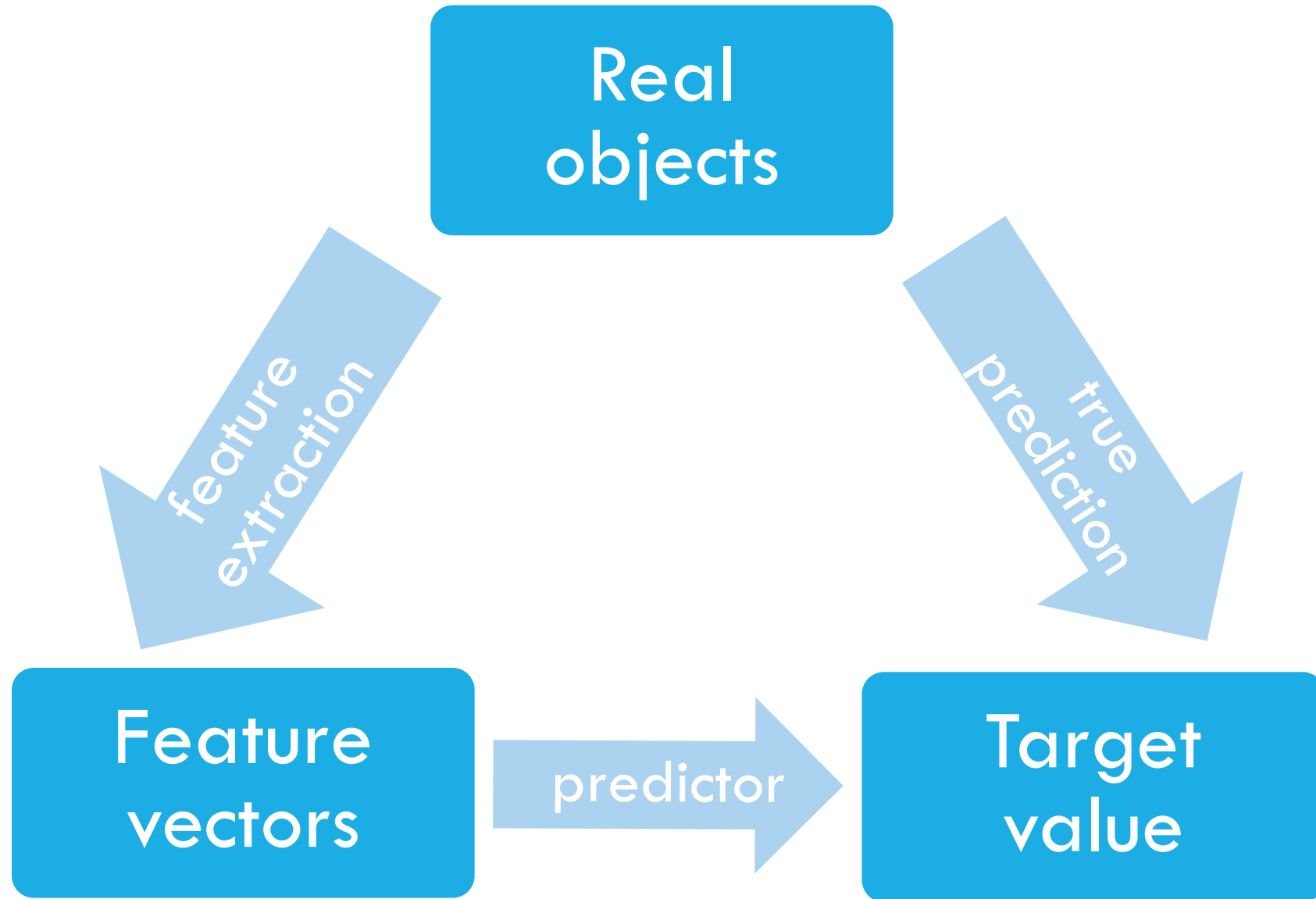
From training data computer learns a **predictor**, a model for new data, representing the “essential knowledge”.

Useful for e.g. **dependency parsing** = guessing the correct dependency tree for a given sentence.

Dependency parsing is a **classification** task: the target values to be guessed are discrete (vs **regression** if continuous).

Features are observable properties of the examples. **Feature vectors** are ordered lists of features.

A data instance is a feature vector paired with a target value. **Training data** is a set of data instances.



Supervised Machine Learning for Dependency Parsing:

Data instances = treebanks, to be divided into training and test sets.

Feature vectors = lemmas, part-of-speech tags, morphological features, etc...

Target values = head and deprel.

What is the essential knowledge a machine should learn for parsing? It depends on the algorithm. When it observes some feature, it should perform the correct action or assign the correct probability to alternative dependencies.

Supervised Machine Learning techniques are language-independent (vs rule-based).

If we can handle complex languages, even more so we can handle other languages in their real manifestations inside texts.

x_i = feature vectors, y_i = true values, z_i = predicted values.

h^* = **prediction function**, the best among the hypotheses H .

Ideally, $z_i = h^*(x_i) = y_i$

The aim is finding h^* , searching through the hypothesis space.

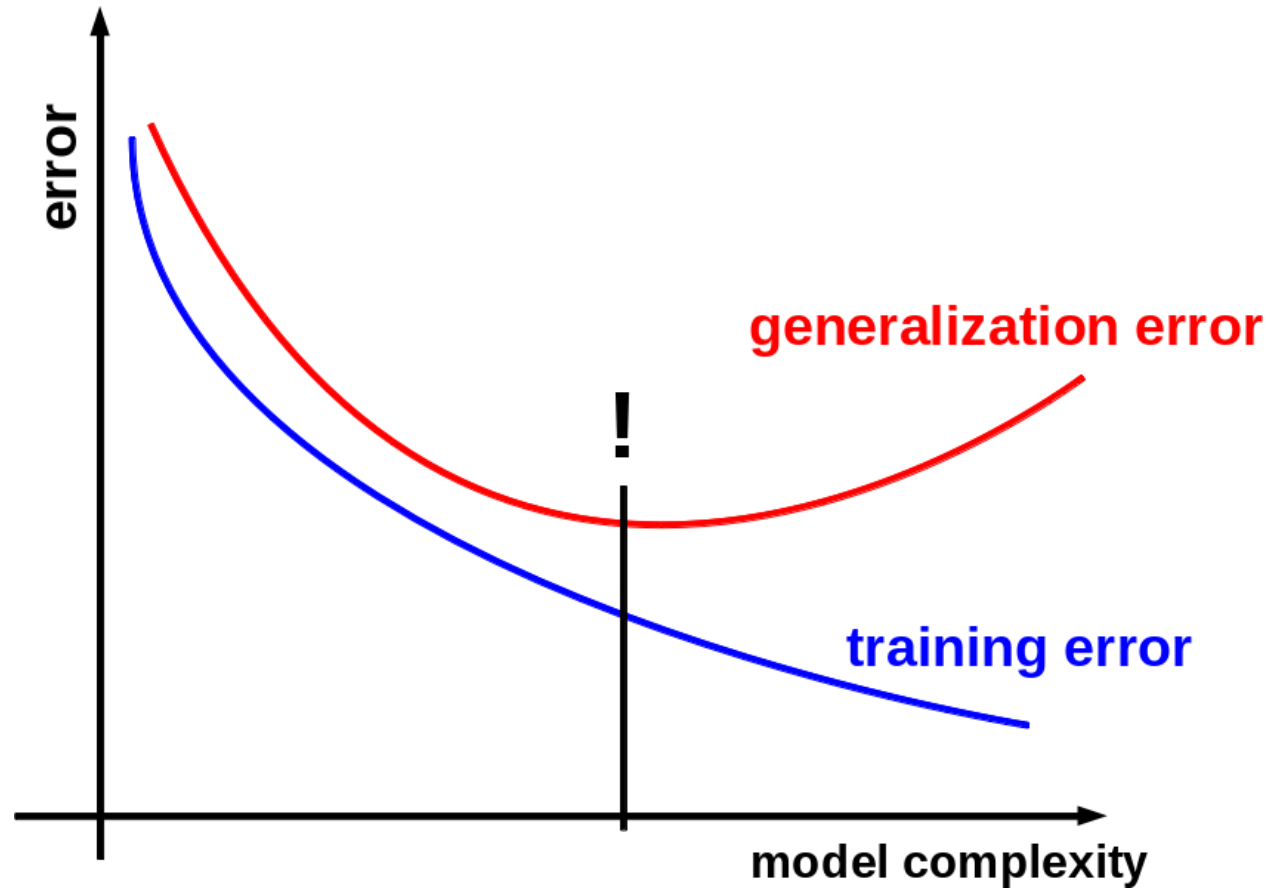
Loss function $L(z, y)$ = cost of predicting z when y is true.

E.g. zero-one loss = $I(z_i \neq y_i)$, $(z_i \neq y_i) \rightarrow I = 1$, $(z_i = y_i) \rightarrow I = 0$

The aim can be restated as minimizing the average loss.

Sample error = average zero-one loss.

Generalization error = how bad h^* generalizes beyond training data to new data.



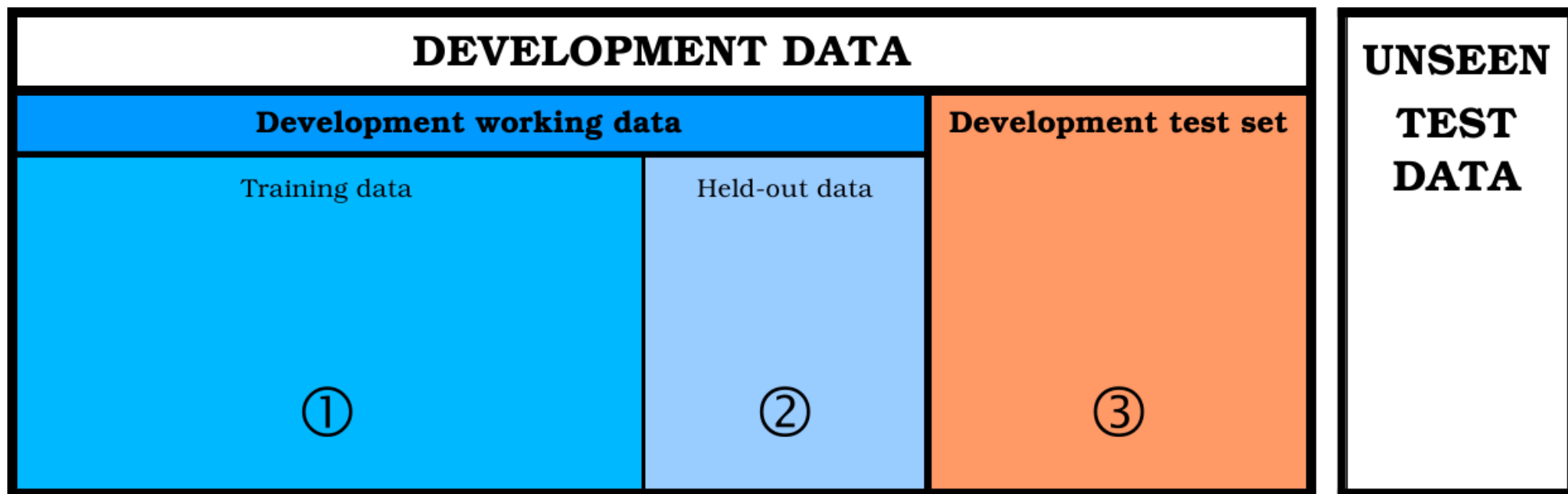
Predictor building involves the choice of algorithm, its parameters and features. After this, a model is created feeding **training data** to the machine.

Then predictor is tested against the **test data**. Randomly chosen from the whole data and distinct from training data.

Final stage is **evaluation**, through comparison of predicted and real values in test data. During this phase, these metrics are used to assess the accuracy:

Labeled Attachment Score (LAS) = correct heads and deprels guessed / total heads and deprels

Unlabeled Attachment Score (UAS) = correct heads guessed / total heads



AN EXAMPLE OF PARSER: DESR

Attardi (2006): deterministic-choice and linear. Trees are built bottom-up, either left-to-right (LR) or right-to-left (RL).

Shift-reduce = algorithm processes each token of the sentence in linear order. Local optimization criterion to fit the data. Best overall performance, but less accurate with complex (e.g. non-projective) structures.

Modular = several learning algorithm available, e.g. Support Vector Machine (SVM), Maximum Entropy (ME), MultiLayer Perceptron (MLP).

State of the parser = $\langle S, I, T, A \rangle$ = stack, remaining tokens, temporary tokens in the stack, arc relation.

W = words of the sentence

Initial state = $\langle (), W, (), () \rangle$

End state = $\langle S, (), (), A \rangle$

Shift = in a configuration $\langle S, n | l, T, A \rangle$, pushes n to the stack, producing the configuration $\langle n | S, l, T, A \rangle$.

Right = in a configuration $\langle s | S, n | l, T, A \rangle$, adds an arc from s to n and pops s from the stack, producing the configuration $\langle S, n | l, T, AU\{(s, r, n)\} \rangle$.

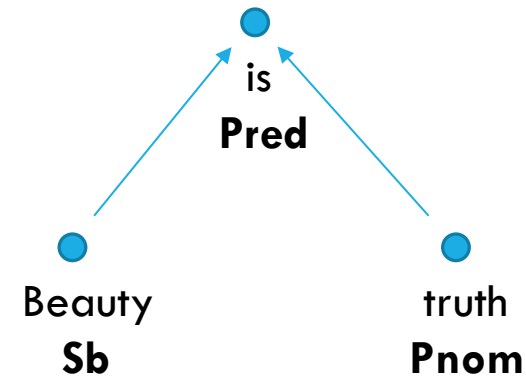
Left = in a configuration $\langle s | S, n | l, T, A \rangle$, adds an arc from n to s , pops n from input, pops s from the stack and moves it back to l , producing the configuration $\langle S, s | l, T, AU\{(n, r, s)\} \rangle$.

Plus, six others to handle non-projectivity...

Essential knowledge consists in performing the correct actions and choosing the correct relation label given an input sentence in a language. Actions and labelling are performed deterministically according to a feature model. Better the model, better the prediction.

A simplified example: parse of «Beauty is truth»

Stack	Input	Action
	Beauty is truth	Shift
Beauty	is truth	Right
	is truth	Shift
is	truth	Left
is		



‘is’ corresponds to a predicate: as it is the root, it means that the sentence is complete.

In other terms, the last state is a final state, hence the sequence is accepted.

The table accounts only for arc creation: labelling takes place simultaneously with Left and Right but here is shadowed for sake of simplicity. Also, I abstracted from the temporary stack column, useful for non-projective trees only.

The model posits as conditions for a given action the presence in input and stack of specific elements with specific properties.

A CASE OF STUDY: MEDIEVAL LATIN

Passarotti & Ruffolo (2010)

Parser	LAS	UAS
DeSR	71,26	78,35
Malt	69,85	75,87
ISBN	68,97	77,79
MST	68,79	79,43

DesR with Italian/Czech feature model

Passarotti & dell'Orletta (2010)

Data set	Tokens
Training	44195
Test	5697

Data set	Tokens
Training	61024
Test	7379

DeSR with ad hoc feature model →

Feature	Tokens
LEMMA	-2 -1 0 1 2 3 <i>prev(0) next(-1) leftChild(-1) leftChild(0) rightChild(-1) rightChild(0)</i>
POSTAG	-2 -1 0 1 2 3 <i>prev(0) next(-1) leftChild(-1) leftChild(0) rightChild(-1) rightChild(0)</i>
CPOSTAG	-1 0 1 2
FEATS	-1 0 1 2
DEPREL	<i>rightChild(-1)</i>
HEAD	-1 0

Filter = manual selection as a preprocessing step, independently by the algorithm.

Usefulness of tailor-made feature model:

Reduces overfitting, hence increases performance.

Reduces model dimension, hence complexity, hence computational time.

Improved model interpretability.

Results on dataset of *Passarotti & Ruffolo (2010)*

Parser	Algorithm	Direction	LAS	UAS
DeSR	SVM*	LR	73,73	79,90

Results on dataset of *Passarotti & dell'Orletta (2010)*

Parser	Algorithm	Direction	LAS	UAS
DeSR	SVM	LR	78,26	83,9
DeSR	SVM	RL	76,31	82,38

Dataset size has a great impact on performance!

* Both experiments use Support Vector Machines (SVMs) as algorithm. SVMs represent examples as points in space, maximizing the geometrical gap among separate classes. For non-binary tasks, they are implicitly mapped to a multi-dimensional space.

POSTPROCESSING TECHNIQUES: REVISION...

Attardi & Dell'Orletta, 2009

Stacked parsing has two phases:

- 1) Sentence is parsed with a low-performing algorithm.
- 2) A second parser with high-performing algorithm learns from the output of 1) in reverse mode (i.e. in opposite direction), using additional features.

They adopted these names for stacked parsers:

Rev2 = 1) ME LR 2) SVM RL

Rev3 = 1) ME RL 2) SVM LR

Parser	Predictor	LAS	UAS
DeSR	Rev2	77,30	82,82
DeSR	Rev3	79,27	84,63

...AND COMBINATION

Consider a set of predictors.

The ensemble of their outputs will outperform the output of a single predictor, given a sufficient accuracy and diversity of the set members.

Ideally, predictors should be statistically independent. But: too much data required.

Post-processing technique. Outputs with predicted values are combined.

Surdeanu and Manning (2010)

The most common function is based on unweighted voting. The value voted by the majority is selected. As accurate as more complex functions.

Linear-time re-parsing algorithms guarantee well-formed dependency trees.

Combined predictors	LAS	UAS
SVM-rev3 + SVM-RL + SVM-LR	80,02	85,23
SVM-rev2 + SVM-rev3 + SVM-RL	79,82	85,08
SVM-rev2 + SVM-rev3 + SVM-LR	79,21	84,67
SVM-rev2 + SVM-RL + SVM-LR	78,91	84,36

This is the state of the art in literature!

NEW IMPROVEMENTS

Key ideas:

- ❖ Combining not different predictors from the same parser, but different parsers.
- ❖ Training over a far richer dataset.
- ❖ Experimenting new feature models.

Parser #2: **MATE graph-based**

Bohnet (2010)

Graph-based parser = each sentence is processed as a whole. A global optimization criterion is adopted to best fit the data.

Multi-digraphs (MDG) are sets of nodes V and oriented arcs A , allowing multiple arcs between any pair of nodes.

A spanning tree (ST) is a subgraph of a MDG, $G' = (V', A')$ such that:

- 1) The set of nodes is identical. $V = V'$
- 2) The set of arcs of MDG includes that of ST. $A \subseteq A'$
- 3) The cardinality of its arcs equals that of its nodes minus 1. $|A'| = |V'| - 1$
- 4) G' is an acyclic graph.

Given an MDG, let T be the set of its STs.

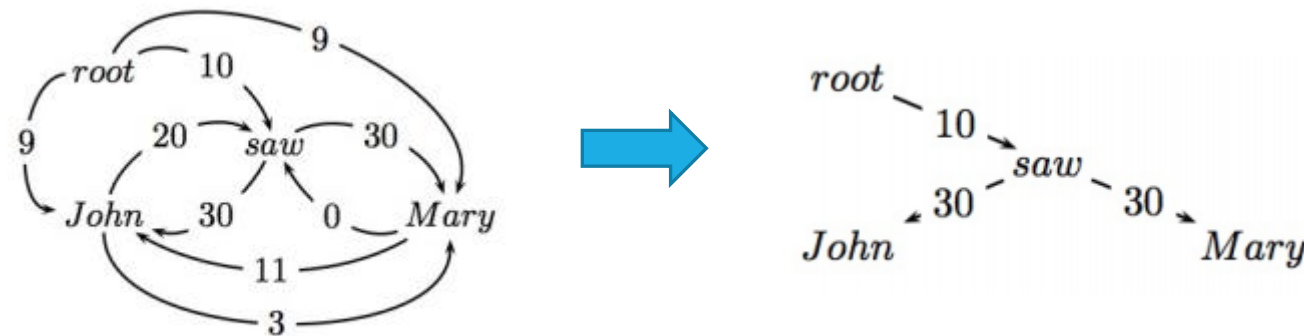
Maximum spanning tree problem is finding the member of T that maximizes a given function.

In dependency parsing terms, given a set of possible trees for an input sentence, a predictor should select the most plausible one.

How is the problem solved? Decomposing the tree into factors and weighting them.

Factor = head, dependent, label. Quadratic complexity.

Example taken from MacDonald and Nivre's slides:



In MATE graph-based parser:

2nd-order maximum spanning tree (MST) + passive-aggressive perceptron.

2nd-order uses more factors (grandchild, label to sisters...). Complexity = $O(n^4)$.

Passive-aggressive perceptron: each iteration weight realignment becomes softer.

Plus, non-projective approximation algorithm and a feature extraction component.

Marries high accuracy with fast parsing.

Algorithm of graph-based MATE-tools

$y = \text{gold}$

for $n \leftarrow 1$ **to** E // iteration over the training epochs

for $i \leftarrow 1$ **to** l // iteration over the training examples

$k \leftarrow (n - 1) * l + i$

$\gamma \leftarrow E * l - k + 2$ // passive-aggressive weights

$A \leftarrow \text{extract-features-\&-calculate-arrays}(i, w)$

$p \leftarrow \text{predict-projective-parse-tree}(A)$

$a \leftarrow \text{non-projective-approximator}(p, A)$

 update w, v according to $\Delta(p, y)$ and γ

$w = v / (E * l)$ // average

Parser #3 = **MATE transition-based**

Bohnet et al. (2013)

Shift-Reduce. Operations Left-Arc, Right-Arc, Shift, Swap (for non-projectivity).

Designed for free-word-order and richly inflected languages.

Joint morphological disambiguation and dependency parsing (vs pipeline).

4 labelling functions for pos, morphological features, lemma and dependency relation

A transition sequence is an ordered set of state-transition pairs.

Candidate parses for a sentences are scored via the transition system leading to them, directed by features and weights.

Uses beam search for exploring the search space (here very large!) of parses, better than greedy deterministic algorithms. In a graph, expands most promising node up to a limit (or up to when gold parse is pruned from the beam in learning).

New (simplified and better-performing) feature model:

Feature	Value
LEMMA	-2 -1 0 1 2 3 <i>prev(0) next(-1) leftChild(-1) leftChild(0) rightChild(-1) rightChild(0)</i>
POSTAG	-2 -1 0 1 2 3 <i>prev(0) next(-1) leftChild(-1) leftChild(0) rightChild(-1) rightChild(0)</i>
CPOSTAG	-1 0 1 2
FEATS	-1 0 1 2
DEPREL	<i>rightChild(-1)</i>
HEAD	-1 0

A wider dataset. *Summa contra gentiles* partes 1-2:

Dataset	Tokens
Training	152210
Test	16935

Results for single predictors:

Parser	LAS	UAS	#
desr LR	79,91	85,97	1a
desr RL	81,36	87,2	1b
mate	76,75	84,41	2
joint (pre-lemmatized)	80,06	86,12	3

Results for combination:

Combination	LAS	UAS
1b+3+1a+2	84,4	89,6
1b+3+1a	83,9	89,2
1b+1a+2	83,3	88,6

New best result!

+4,4% in LAS

+4,4% in UAS

compared with state of art.

BIBLIOGRAPHY

- Rosa, R., Mašek, J., Mareček, D., Popel, M., Zeman, D., & Žabokrtský, Z. (2014, May). HamleDT 2.0: Thirty dependency treebanks stanfordized. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)* (pp. 2334-2341).
- Mambrini, F., Passarotti, M. (2013). *Non-projectivity in the Ancient Greek Dependency Treebank*. DepLing 2013, 177.
- Attardi, G. (2006, June). Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning* (pp. 166-170). Association for Computational Linguistics.
- Surdeanu, M., & Manning, C. D. (2010, June). Ensemble models for dependency parsing: cheap and good?. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 649-652). Association for Computational Linguistics.

- ❖ Attardi, G., & Dell'Orletta, F. (2009, May). Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers* (pp. 261-264). Association for Computational Linguistics.
- ❖ Passarotti, M. C., & Ruffolo, P. (2010). Parsing the Index Thomisticus Treebank. Some preliminary results. In *15th International Colloquium on Latin Linguistics* (pp. 714-725). Innsbrucker Beiträge zur Sprachwissenschaft.
- ❖ Passarotti, M., & Dell'Orletta, F. (2010). Improvements in parsing the index thomisticus treebank. Revision, combination and a feature model for medieval Latin. *Training*, 2, 61-024.
- ❖ Bohnet, B. (2010, August). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 89-97). Association for Computational Linguistics.
- ❖ Bohnet, B. (2010, August). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 89-97). Association for Computational Linguistics.

- ❖ Cer, D. M., De Marneffe, M. C., Jurafsky, D., & Manning, C. D. (2010, May). Parsing to Stanford Dependencies: Trade-offs between Speed and Accuracy. In *LREC*.
- ❖ Kahane, Sylvain. "Why to choose dependency rather than constituency for syntax: a formal point of view." *J. Apresjan, M.-C. L'Homme, L. Iomdin, J. Milićević, A. Polguère, L. Wanner, eds., Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel'čuk, Languages of Slavic Culture, Moscow (2012): 257-272.*

