Today: Week 6, lecture
Today’s topic: Syntactic Analysis
Today’s teacher: Daniel Zeman
E-mail: zeman@ufal.mff.cuni.cz
WWW: http://ufal.mff.cuni.cz/daniel-zeman
Level of (Surface) Syntax

- Relations between sentence parts
- Sentence part = token (word, number, punctuation)
  - Practical reasons:
    - Easily recognizable.
    - Unit of previous (morphological) level of processing.
    - We don’t restore elided constituents, nor do we collapse nodes of function words; this can be done later on a deep-syntactic level.
  - On the other hand:
    - We must now also define relations between function words (prepositions, auxiliary verbs etc.), punctuation and the rest of the sentence.
Level of Surface Syntax

• Between morphology and meaning.

• Morphology provides / requires:
  – lemmas (it’s time to obtain syntactic info from the dictionary)
  – tags (part of speech and morphosyntactic features)
  – word order (now it starts to play a role)

• Typical input is ambiguous
  – ambiguous morphological analysis

• Typical output is ambiguous
  – several syntactic structures for one sentence (several readings of the sentence)
Syntactic Structure

• Different shapes in different theories

• Typically a tree
  – Phrasal (constituent) tree, parse tree

  – Dependency tree
Example of Constituent Tree

- \(((\text{Paul} \ (\text{gave} \ \text{Peter} \ (\text{two} \ \text{pears})))) \ .\)
Example of Dependency Tree

- [#,0] (gave,2) ([Paul,1], [Peter,3], [pears,5] ([two,4])), [.,6])

```
#
|
|
|
gave
|
|
Paul
|
|
Peter
|
|
pears
|
|
two
```

Paul gave Peter two pears.
Words and Phrases

• **Word (token)**
  - smallest unit of the syntactic layer
  - grammatical (function, synsemantic) words (e.g. *and* in coordination *Paul and Peter, to be* in compound verb forms *he is scared, he will be scared*)
  - lexical (content, autosemantic) words (e.g. *dog; to be* in the sentence *I think, therefore I am.* (René Descartes))

• **Phrase**
  - composed of words and/or other phrases (**immediate constituents**)
Words

• Relation to other words
  – Lexicon contains information on words and possible relations among them.
    • Subcategorization of verbs and other words (do they require an object? if so, should it be marked for a particular case?)
    • Semantic features (a noun has color, has size, can act as the subject of a particular set of verbs…)

• Idioms, multi-word expressions
  – Fixed, indivisible phrases may act as one word (e.g. compound prepositions (in spite of), foreign citations and named entities (Rio de Janeiro), compound nouns written as separate tokens (stock exchange))
Phrase Replaceability

• A phrase can be replaced by another phrase of the same type. Specifically, it can be replaced by its head.
  – This is related to the generation of the sentence.

⇒ The phrases \( x, y, z \) can be immediate constituents of a larger phrase \( f \) only if they are related to each other. This is however a matter of the particular phrase structure grammar.
  – Example: sentence “This is the man that I talked about.” The part “man that I” is not a whole noun phrase because it cannot be replaced by another noun phrase, e.g. man: “*This is the man talked about.”"
Phrase

• Phrase
  – Sequence of immediate constituents (words or phrases).
  – May be discontinuous in some languages. cs: „Soubor se nepodařilo otevřít.“ (lit. File oneself one-was-not-able to-open) contains the phrase “open file”.

• Phrase types by their main word—head
  – Noun phrase: the new book of my grandpa
  – Adjectival phrase: brand new
  – Adverbial phrase: very well
  – Prepositional phrase: in the classroom
  – Verb phrase: to catch a ball
Noun Phrase

• A noun or a (substantive) pronoun is the head.
  – water
  – the book
  – new ideas
  – two millions of inhabitants
  – one small village
  – the greatest price movement in one year since the World War II
  – operating system that, regardless of all efforts by our admin, crashes just too often
  – he
  – whoever
Adjective Phrase

• An adjective or a determiner (attributive pronoun) is the head.
• Simple ADJs are very frequent, complex ones are rare.
  – old
  – very old
  – really very old
  – five times older than the oldest elephant in our ZOO
  – sure that he will arrive first
Pronouns / Determiners

• (Substantive) pronouns: similar behavior as nouns
  – Personal pronouns (*I, you, they, oneself*).
  – Some demonstrative, interrogative, relative and negative (*who, what, somebody, something, nothing*).

• Attributive pronouns (determiners): similar behavior as adjectives
  – Possessive pronouns (*my, your, his, whose*).
  – Articles (*the, a, an*).
  – Attributively used demonstrative, interrogative, relative and negative pronouns (*which, some, every, no*).
Numeral Phrases

• In Slavic languages not always clear what should be the head: the number, or the counted noun phrase?
  – The numeral inherits the gender of the counted noun. The noun gets its grammatical number from the numeral.
    • *jeden muž* (one man), *jedna žena* (one woman), *jedno dítě* (one child)
    • *dva muži* (two men), *dvě ženy* (two women), *dvě děti* (two children)
  – The numeral governs the case of the counted noun.
    • *pět mužů* (five men: noun in genitive, numeral in nominative, accusative or vocative)
  – Both the counted noun and the numeral have a case required by their governing preposition or verb.
    • *pěti ženami* (five women: instrumental)
Adverbial Phrases

• An adverb is the head.
  – quickly
  – much more
  – how
  – louder than you can imagine
  – yesterday
Prepositional (Postpositional) Phrase

• The preposition serves as head (because it determines the case of the rest of the phrase).
• Often have a function similar to adverbial phrases (adverbiale) or noun phrases (object of a verb).
  – *in the city center*
  – *in God*
  – *around five o’clock*
  – *to a better future*
  – *up to a situation where neither of them could back out*
  – *with respect to his nonage*
Prepositional Phrases

• Classic English example:
  – *I saw the man with a telescope.*
    1. Viděl jsem ho dalekohledem.
    2. Viděl jsem ho s dalekohledem.
Prepositional Phrases: Czech Example

• „Přišel ten pán se sousedem odnaproto.“

Lit.: Came the man with neighbor from-across-the-road.
Prepositional Phrases and Syntactic Ambiguities

• *V letech 1991 – 1993 jsem absolvovala kurzy řízení a marketingu na Collège Bart v kanadském Québecu.*

• *In years 1991 – 1993 I attended classes of management and marketing at Collège Bart in Canadian Québec.*

(A Czech sentence from the Prague Dependency Treebank.)
Prepositional Phrases and Syntactic Ambiguities

- In years 1991 – 1993 I attended classes of management and marketing at Collège Bart in Canadian Québec.

  - attended at Collège Bart
  - classes at Collège Bart
  - management and marketing at Collège Bart
  - marketing at Collège Bart
  - Collège Bart in Québec
  - marketing in Québec...
Prepositional Phrases and Syntactic Ambiguities

• In years 1991 – 1993 I attended classes of management and marketing at Collège Bart in Canadian Québec.

  – attended (class (of (mngmt and market))) (at Bart)
  – attended (class (of (mngmt and market)) (at Bart))
  – attended (class (of ((mngmt and market) (at Bart))))
  – attended (class (of (mngmt and (market (at Bart)))))
  – … ((at Bart) (in Québec))
  • Is Bart in Québec or Québec in Bart?
Prepositional Phrases and Syntactic Ambiguities

• „říjnové jednání OSN o klimatických změnách v Kodani“ (Události ČT, 27.2.2009)
• “October UNO summit about climatic changes in Copenhagen” (Czech TV news, 2-27-2009)

• Question:
Were there climatic changes in Copenhagen?
Verb Phrase

• The underlined finite verb form is the head.
• The repertory depends on the rules for analytical verb forms and varies greatly cross-linguistically.

  – it **rains**
  – he **could** at all sight Mr. President
  – why we **got** wet so much
  – **Go!**
  – he **has been transported** to the hospital on Sunday
  – it **began** to rain
  – **prohibits** smoking in this room
  – give Mary the beads that we **brought** from the vacation in Morocco
  – the file **could** not be opened
Clause

• Group of words with 1 predicate, e.g.:
  – *John loves Mary.*
  – …*that you are right.*

• Not necessarily same as a verb phrase (VP).
  – Nested VPs are part of the main VP.
  – Nested clauses are not parts of the main clause.
Clause and Sentence

• Clause
  – simple sentence or part of compound sentence
  – e.g. *John loves Mary.* or “*that you are right*”.

• Sentence
  – simple sentence or compound sentence
  – consists of one or more clauses
  – e.g. *John loves Mary.* or “*I realized that you were right.*”
Clause

• Predicative function
  – Certain activity of certain subjects and objects in certain time under certain conditions

• Main clause
  – Independent of other clauses in the sentence

• Nested clause, relative clause
  – Depends on another clause, carries out a function in that clause (as a dependent phrase)

• Functions of clauses:
  – Same as phrases plus some special, e.g. direct speech.
Sentence

- Consists of one or more main clauses.
- If there are more than one main clause then they are usually coordinated.
- A written sentence begins with a capital letter (if the script distinguishes case). Sometimes begins with a parenthesis or a quotation mark. An uppercase letter can occur inside of the sentence, too.
- It ends with a period, exclamation or question mark. Sometimes ends with a parenthesis or a quotation mark. A period can occur inside of the sentence, too.
- Depending on human decision, semicolons and colons may or may not terminate a sentence. It is usually possible to view them as coordinating conjunctions.
Coordination

• There is no real head. Technically, the conjunction, comma etc. can be proclaimed a head.
• The coordinated phrases are usually of the same type.
  – chickens, hens, rabbits, cats and dogs
  – new or even newer
  – quickly and finely
  – he came to the conclusion that there is no point in hiding any more, so we might hear him here today
  – in the house or outside
  – to and from Prague
  – either now or later
  – not only on Monday and on Wednesday but also tomorrow or the day after tomorrow
Apposition

- Similarly to coordination, joins two phrases none of which depends on the other.
- Unlike coordination, apposition has never more than two members.
- The combined meaning is also different:
  - *Charles IV, Roman Emperor and Czech King*
- Coordination: multiple different phrases carry out the same function together.
- Apposition: semantically only one entity; on surface, it is described by two different ways.
  - *and the most — 40 percent — befalls to family homes*
  - *factors, especially depreciation*
  - *caretaker — natural or legal person determined by the owner of the building*
  - *costs and increase of taxes — these are matters that...*
Elision

• A phrase omitted from the (surface of the) sentence although it is present in the underlying meaning (deep structure).
• Frequently in dialogues: the elided phrase is known from context.
  – Whom did you see there? — Peter. (Missing verb.)
• In written text often occurs in coordination.
  – Czech and German researchers discussed… (There was probably no researcher that was Czech and German at the same time. Instead, there were Czech researchers and German researchers.)
  – The Penguins are leading 4:0, while the Colorado Avalanches only 3:2. (verb in the second part)
• Systemic elision of subject in pro-drop languages (it is marked on the verb and can be deduced in the form of a pronoun).
  – Sedím. (já) = “(I) sit.”
Gaps and Discontinuous Phrases

• A constituent (phrase) was moved from the position where it is expected.
• Nothing special in free-word-order languages. The terms *gap* and *trace* are typically used in English (see the Penn Treebank).
• In Czech: *gap* is a term related to non-projective constructions and its meaning is different!
• English questions and relative clauses:
  – *Who do you work for* <gap> *whom*?
  – *I don’t know why we have got so much rain* <gap> *why*.
  – *On Sundays, I usually work* <gap> *on sundays but I stay at home on Tuesdays.*
  – *the story he never wrote* <gap> *the story*
Summary of Phrase-Based Model

- Sentence is divided to phrases (constituents).
- Phrase may be divided to even smaller phrases.
- The largest phrase is the whole sentence.
- The smallest phrase is a word.
- Phrases are named and labeled according to their type.
Observation: Phrases Are Related to Context-Free Grammars

- Phrase structure of a sentence corresponds to the derivation tree under the grammar that generates / recognizes the sentence.
- Example:
  - $S \rightarrow NP \ VP$ (a sentence has a subject and a predicate)
  - $NP \rightarrow N$ (a noun is a noun phrase)
  - $VP \rightarrow V \ NP$ (a verb phrase consists of a verb and its object)
- Lexicon part of the grammar:
  - $N \rightarrow$ dog | cat | man | car | John …
  - $V \rightarrow$ see | sees | saw | bring | brings | brought | …
Lexicon

• In practice the lexical part can (and should) be implemented separately from the grammar.

• The nonterminals of the lowest level (immediately above the terminals) might be POS tags.
  – Then morphological analysis and tagging (disambiguation of MA) solves the lowest level of the phrase tree.
    • In fact, disambiguation is not necessary. There will be other ambiguities in the tree anyway. The parser can take care of them.
  – The grammar works only with POS tags.
  – This is why we sometimes talk about preterminals (the nonterminals immediately above the leaf nodes).
An Extended Grammar Example for Czech (7 Cases!)

- NP → N | AP N
- AP → A | AdvP A
- AdvP → Adv | AdvP Adv

- NP_{nom} → N_{nom}
- NP_{nom} → AP_{nom} N_{nom}
- NP_{nom} → N_{nom} NP_{gen}

- NP_{gen} → N_{gen}
- NP_{gen} → AP_{gen} N_{gen}
- NP_{gen} → N_{gen} NP_{gen}

- N → pán | hrad | muž | stroj ...
- A → mladý | velký | zelený ...
- Adv → velmi | včera | zeleně ...

- N_{nom} → pán | hrad | muž ...
- N_{gen} → pána | hradu | muže ...
- N_{dat} → pánovi | hradu | muži ...
- N_{acc} → pána | hrad | muže ...
- N_{voc} → pane | hrade | muži ...
- N_{loc} → pánovi | hradu | muži ...
- N_{ins} → pánem | hradem ...

9.12.1999
http://ufal.mff.cuni.cz/course/npfl094
An Extended Grammar Example for Czech (Verbs)

- \( VP \rightarrow VP_{\text{obligatory}} \)
- \( VP \rightarrow VP_{\text{obligatory}} \ VP_{\text{optional}} \)
- \( VP_{\text{obligatory}} \rightarrow V_{\text{intr}} \)
- \( VP_{\text{obligatory}} \rightarrow V_{\text{trans}} \ NP_{\text{acc}} \)
- \( VP_{\text{obligatory}} \rightarrow V_{\text{bitr}} \ NP_{\text{dat}} \ NP_{\text{acc}} \)
- \( VP_{\text{obligatory}} \rightarrow V_{\text{mod}} \ V_{\text{INF}} \)
- \( VP_{\text{optional}} \rightarrow \text{AdvP}_{\text{location}} \ | \ \text{AdvP}_{\text{time}} \ ... \)

- \( V_{\text{intr}} \rightarrow šedivět \ | \ brzdit \ ... \)
- \( V_{\text{trans}} \rightarrow koupit \ | \ ukrást \ ... \)
- \( V_{\text{bitr}} \rightarrow dát \ | \ půjčit \ | \ poslat \ ... \)
- \( V_{\text{mod}} \rightarrow moci \ | \ smět \ | \ muset \ ... \)

- \( ... \) (tens to hundreds of frames)
Unification Grammar

• An alternative to nonterminal splitting
• Instead of seven context-free rules:
  – $NP_{nom} \rightarrow AP_{nom} N_{nom}$
  – $NP_{gen} \rightarrow AP_{gen} N_{gen}$
  – $NP_{dat} \rightarrow AP_{dat} N_{dat}$
  – $NP_{acc} \rightarrow AP_{acc} N_{acc}$
  – $NP_{voc} \rightarrow AP_{voc} N_{voc}$
  – $NP_{loc} \rightarrow AP_{loc} N_{loc}$
  – $NP_{ins} \rightarrow AP_{ins} N_{ins}$
• One unification rule:
  – $NP \rightarrow AP N := \text{[case} = \text{AP}^\text{case} \# \text{N}^\text{case}]$
Syntactic Analysis (Parsing)

• Automatic methods of finding the syntactic structure for a sentence
  – Symbolic methods: a phrase grammar or another description of the structure of language is required. Then: the chart parser.
  – Statistical methods: a text corpus with syntactic structures is needed (a treebank).
  – Hybrid methods: a simple grammar, ambiguities solved statistically with a corpus.
• Chunking / shallow parsing
Parsing with a Context-Free Grammar

• Hierarchy of grammars:
  – Noam Chomsky (1957): *Syntactic Structures*

• Couple of classical algorithms.
  – CYK (Cocke-Younger-Kasami) … complexity $O(n^3)$
    • John Cocke ("inventor")
    • Tadao Kasami (1965), Bedford, MA, USA (another independent "inventor")
    • Daniel H. Younger (1967) (computational complexity analysis)

• Constraint of CYK: grammar is in CNF (Chomsky Normal Form), i.e. the right-hand side of every rule consists of either two nonterminals or one terminal. (CFGs can be easily transformed to CNF.)
Parsing with a Context-Free Grammar

- **Chart parser**: CYK requires a data structure to hold information about partially processed possibilities. Turn of 1960s and 1970s: the chart structure proposed for this purpose.

- Jay Earley (1968), PhD thesis, Pittsburgh, PA, USA
  - A somewhat different version of chart parsing.

- For details on chart parser, see the earlier lecture about morphology and context-free grammars.
Practical Phrase-Based Parsing

• Rule-based parsers, e.g. Fidditch (Donald Hindle, 1983)

• **Collins** parser (Michael Collins, 1996–1999)
  – Probabilistic context-free grammars, lexical heads
  – Labeled precision & recall on Penn Treebank / Wall Street Journal data / Section 23 = 85%
  – Reimplemented in Java by Dan Bikel ("Bikel parser"), freely available

• **Charniak** parser (Eugene Charniak, NAACL 2000)
  – Maximum entropy inspired parser
  – P ~ R ~ 89.5%
  – Mark Johnson: reranker => over 90%

• **Stanford** parser (Chris Manning et al., 2002–2010)
  – Produces dependencies, too. Initial P ~ R ~ 86.4%
Dependency Parsing

Daniel Zeman

http://ufal.mff.cuni.cz/daniel-zeman/
Dependency Model of Syntax

• Summary of syntactic relations:

• Sentence divided to phrases (constituents).
  – Cornerstone of the phrase-based (constituent-based) model.

• Phrase head, dependency of other phrase members on the head.
  – Head = governing node (token), the other nodes are dependent.
  – Cornerstone of a dependency tree.

• We can talk of dependencies even if we work with constituent trees and vice versa.
Example of Dependency Tree

- [#,0] ([gave,2] ([Paul,1], [Peter,3], [pears,5] ([two,4]))), [.,6])

```
#   gave
   /   \\
Paul Peter pears
   /     /   \\
   two
```
Dependency Labels

Paul / Sb
gave / Pred
Peter / Obj
pears / Obj

two / Atr

# / AuxS
gave / Pred
.


45
Phrase vs. Dependency Trees

Paul gave Peter two pears.
Phrase vs. Dependency Trees

• Phrase (constituent) trees
  – Show decomposition of sentence to phrases and label them.
  – Don’t stress what is head and which word depends on which.
  – Needn’t specify function, dependency type.

• Dependency trees
  – Show dependencies between words and label them.
  – Don’t capture similarity of construction of different sentence parts, recursion.
  – Don’t capture progress of sentence generation, proximity of dependent nodes to the head.
  – Don’t contain nonterminals, phrase types—these can be only estimated from parts of speech of the heads.
Differences between Phrase and Dependency Model

- We want to convert a phrase tree $P$ to a dependency tree $D$ or vice versa.
- Phrase tree does not tell what is the phrase head.
  - To convert $P \rightarrow D$ we need a selection function that for every grammar rule select a right-hand symbol to serve as the head.
- Dependency tree does not show how the sentence arose (recursion), nor does it necessarily cover the complete phrase decomposition.
  - It does not tell what has been added “sooner” and what “later”.
  - Several phrase structures may lead to the same dependency structure $\Rightarrow$ back conversion $(D \rightarrow P)$ is ambiguous.
Example

- Several phrase trees lead to the same dependency tree.

```
S(bought)                      S(bought)
  NP(John)   VP(bought)        VP(bought)
    V(bought) NP(bike)         NP(John) V(bought) NP(bike)
```

bought  bought
John    bike
Differences between Phrase and Dependency Model

- Dependency tree does not know phrase labels (nonterminals—because it does not even know what the phrases are, see previous slide).
  - We need a function that determines the label according to the phrase head.
  - Really we need it? To understand the meaning, one needs the relations and their type but not what has been generated sooner and what later.
- Phrase tree does not know the type of the relation between the head and the other members—function. (But cf. functional tags in Penn Treebank.)
  - We need a function that determines the dependency label for every non-head member of the phrase. (We can tell that while selecting the head.)
- A significant difference: phrase trees are tightly bound to the word order!
Discontinuous Phrases

- Classical context-free grammar cannot describe them!
- They cannot be represented by bracketing.
- *(Soubor (se nepodařilo) otevřít). (cs: File couldn’t be opened)*

```
N(soubor) T(se) V(nepodařilo) VP(nepodařilo)

VR(nepodařilo) VP_inf(otevřít)

N(soubor) T(se) V(nepodařilo) V_inf(otevřít)
```
Nonprojectivity

- Dependency tree including word order (horizontal coordinate of nodes).
- Projection to the base: the vertical from the node crosses a dependency (nonprojective edge).
- Formally:
  - Dependency \( ([g, x_g], [d, x_d]) \). \( x_w \) is the order of the word \( w \) in the sentence.
  - There exists a node \( [n, x_n] \) that \( x_g < x_n < x_d \) or \( x_d < x_n < x_g \) and \( [n, x_n] \) is not in subtree rooted by \( [g, x_g] \).
- Informally: The string spanned by the subtree of the governing node is discontinuous, contains gaps.
Nonprojectivity: Can Be Handled by a Dependency Tree!
Problem: Not Everything is Dependency

• Coordination and apposition.
  – Modifying coordination × modifying a coordination member.
  – Auxiliary nodes (punctuation etc.)
Prepositional Phrases, Nested Subjoined Clauses

budu 
Pred

na
AuxP

zápraží
Adv

na
AuxP

rozdíl
AuxP

Pavla
Adv

od
AuxP

ptáte
Pred

se
AuxT

zda
AuxC

vidím
Obj

vás
Obj
Nested Relative Clauses

muž / man

???

kterého
/ whom

Obj,

AuxX

jsem
/ I

AuxV

vám
/ to you

Obj

představil / introduced

Atr

, kterého jsem vám

AuxX / whom / I / to you

Obj AuxV Obj
Phrases, Dependencies and Other Models

• Phrases (constituents, immediate constituents).
  – Originally more widespread, suitable for English.
  – Context-free grammars.

• Dependencies.
  – Originally popular e.g. in Czech (and also in Far East), now widespread.
  – Especially suitable for free-word-order languages.
  – Dependency grammars, grammars of dependency trees.

• Categorial grammars.
• Tree-adjoining grammars (TAGs).
• And many more…
Dependency Grammar

- In contrast to phrase model, relation to grammar is artificial ("dependency tree does not demonstrate how it was generated").
- No implementation for Czech.
- Context-free grammar + head-selection function (only projective constructions).
- Grammar rules that rewrite a nonterminal to a whole subtree (grammar of dependency trees).
- Related to link grammars, tree-adjoining grammars, categorial grammars.
- HPSG, unification.
MST Parser

• McDonald et al., HLT-EMNLP 2005
• http://sourceforge.net/projects/mstparser/
• MST = maximum spanning tree = ČS: nejlépe ohodnocená kostra (orientovaného) grafu
• Start with a total graph.
  – We assume that there can be a dependency between any two words of the sentence.
• Gradually remove poorly valued edges.
• A statistical algorithm will take care of the valuation.
  – It is trained on edge features.
  – Example features: lemma, POS, case… of governing / dependent node.
MST Parser

• Feature engineering (tell the parser what features to track) by modifying the source code (Java).
• Not easy to incorporate 2nd order features
  – I.e. edge weight depends e.g. on POS tag of its grandparent.
• Parser can be run in nonprojective mode.
• Training on the whole PDT reportedly takes about 30 hours.
  – It is necessary to iterate over all feature combinations and look for the most useful ones.
• In comparison to that, the parsing proper is quite fast.
Malt Parser

• Nivre et al., *Natural Language Engineering*, 2007
• [http://maltparser.org/](http://maltparser.org/)
• Based on *transitions* from one configuration to another.
• Configuration:
  – Input buffer (words of the sentence, left-to-right)
  – Stack
  – Output tree (words, dependencies and dependency labels)
• Transitions:
  – Shift: move word from buffer to stack
  – Larc: left dependency between two topmost words on stack
  – Rarc: right dependency between two topmost words on stack
Malt Parser

- Parser driven by oracle that selects the transition operation based on the current configuration.
- Training: decompose the tree from training data to a sequence of configurations and transitions
  - Sometimes there are more than one possibility
    - Various learning strategies: e.g. create dependencies eagerly, as soon as possible.
- The oracle learns based on the features of the configuration.
  - E.g. word, lemma, POS, case, number...
    - $n^{th}$ word from the top of the stack
    - $k^{th}$ word remaining in the buffer
    - particular node in output tree part created so far
Malt Parser

• Again, a machine learning algorithm is responsible for training, here the Support Vector Machines (SVM).
  – Classifier. Input vectors: values of all features of the current configuration.
  – In addition, during training there is the output value, i.e. action identifier (Shift / Larc / Rarc).
  – The trained oracle (SVM) tells the output value during parsing.
• Training on the whole PDT may take weeks!
  – Complexity $O(n^2)$ where $n$ is number of training examples.
  – Over 3 million training examples can be extracted from PDT.
• Parsing is relatively faster (~ 1 sentence / second) and can be parallelized.
Example of Malt Parsing

- **stack** = #
- **buffer** = Pavel dal Petrovi dvě hrušky .
- **English** = *Paul gave to-Peter two pears* .
Example of Malt Parsing

- stack = #
- buffer = Pavel dal Petrovi dvě hrušky .
- tree =

**SHIFT**

- stack = # Pavel
- buffer = dal Petrovi dvě hrušky .
- tree =
Example of Malt Parsing

- stack = # Pavel
- buffer = dal Petrovi dvě hrušky.
- tree =

**SHIFT**

- stack = # Pavel dal
- buffer = Petrovi dvě hrušky.
- tree =

http://ufal.mff.cuni.cz/course/npfl094
Example of Malt Parsing

- stack = # Pavel dal
- buffer = Petrovi dvě hrušky .
- tree =

LARC

- stack = # dal
- buffer = Petrovi dvě hrušky .
- tree = dal(Pavel)
Example of Malt Parsing

- stack = # dal
- buffer = Petrovi dvě hrušky .
- tree = dal(Pavel)

**SHIFT**

- stack = # dal Petrovi
- buffer = dvě hrušky .
- tree = dal(Pavel)
Example of Malt Parsing

- stack = # dal Petrovi
- buffer = dvě hrušky .
- tree = dal(Pavel)

RARC

- stack = # dal
- buffer = dvě hrušky .
- tree = dal(Pavel,Petrovi)
Example of Malt Parsing

- stack = # dal
- buffer = dvě hrušky .
- tree = dal(Pavel, Petrovi)

**SHIFT**

- stack = # dal dvě
- buffer = hrušky .
- tree = dal(Pavel, Petrovi)
Example of Malt Parsing

• stack = # dal dvě
• buffer = hrušky .
• tree = dal(Pavel,Petrovi)

SHIFT

• stack = # dal dvě hrušky
• buffer = .
• tree = dal(Pavel,Petrovi)
Example of Malt Parsing

- stack = # dal dvě hrušky
- buffer = .
- tree = dal(Pavel,Petrovi)

**LARC**

- stack = # dal hrušky
- buffer = .
- tree = dal(Pavel,Petrovi), hrušky(dvě)
Example of Malt Parsing

- stack = # dal hrušky
- buffer = .
- tree = dal(Pavel,Petrovi),hrušky(dvě)

RARC

- stack = # dal
- buffer = .
- tree = dal(Pavel,Petrovi,hrušky(dvě))
Example of Malt Parsing

- stack = # dal
- buffer = .
- tree = dal(Pavel, Petrovi, hrušky(dvě))

RARC

- stack = #
- buffer = .
- tree = #(dal(Pavel, Petrovi, hrušky(dvě)))
Example of Malt Parsing

- stack = #
- buffer = .
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

SHIFT

- stack = #.
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))
Example of Malt Parsing

- stack = #
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

RARC

- stack = #
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)),.)
Nonprojective Mode of Malt

• It can be proved that the above transition system is
  – correct
    • resulting graph is always a tree (continuous, cycle-free)
  – complete for the set of projective trees
    • every projective tree can be expressed as a sequence of transitions
• How to add nonprojective dependencies?
  – New transition operation **SWAP**:  
  – Take second topmost word from stack and return it to buffer. That will swap the order of the input words.
  – This action is permitted only for words that have not been swapped before (their order on the stack corresponds to their original order in the sentence).
Nonprojective Parsing Example

- stack = #
- buffer = Soubor se nepodařilo otevřít.
- English = File itself it-did-not-succeed to-open.
Nonprojective Parsing Example

- stack = #
- buffer = Soubor se nepodařilo otevřít .
- tree =

SHIFT

- stack = # Soubor
- buffer = se nepodařilo otevřít .
- tree =
Nonprojective Parsing Example

- stack = # Soubor
- buffer = se nepodařilo otevřít.
- tree =

SHIFT

- stack = # Soubor se
- buffer = nepodařilo otevřít.
- tree =
Nonprojective Parsing Example

- stack = # Soubor se
- buffer = nepodařilo otevřít .
- tree =

SHIFT

- stack = # Soubor se nepodařilo
- buffer = otevřít .
- tree =
Nonprojective Parsing Example

- **stack** = # Soubor se nepodařilo
- **buffer** = otevřít .
- **tree** =

*LARC*

- **stack** = # Soubor nepodařilo
- **buffer** = otevřít .
- **tree** = nepodařilo(se)
Nonprojective Parsing Example

• stack = # Soubor nepodařilo
• buffer = otevřít .
• tree = nepodařilo(se)

SHIFT

• stack = # Soubor nepodařilo otevřít
• buffer = .
• tree = nepodařilo(se)
Nonprojective Parsing Example

- stack = # Soubor nepodařilo otevřít
- buffer = .
- tree = nepodařilo(se)

**SWAP**

- stack = # Soubor otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se)
Nonprojective Parsing Example

• stack = # Soubor otevřít
• buffer = nepodařilo .
• tree = nepodařilo(se)

LARC

• stack = # otevřít
• buffer = nepodařilo .
• tree = nepodařilo(se), otevřít(Soubor)
Nonprojective Parsing Example

- stack = # otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se),otevřít(Soubor)

**SHIFT**

- stack = # otevřít nepodařilo
- buffer = .
- tree = nepodařilo(se),otevřít(Soubor)
Nonprojective Parsing Example

• stack = # otevřít nepodařilo
• buffer = .
• tree = nepodařilo(se), otevřít(Soubor)

LARC

• stack = # nepodařilo
• buffer = .
• tree = nepodařilo(se, otevřít(Soubor))
Nonprojective Parsing Example

- stack = # nepodařilo
- buffer = .
- tree = nepodařilo(se, otevřít(Soubor))

RARC

- stack = #
- buffer = .
- tree = #(nepodařilo(se, otevřít(Soubor)))
Nonprojective Parsing Example

- stack = #
- buffer = .
- tree = #(nepodařilo(se, otevřít(Soubor)))

**SHIFT**

- stack = # .
- buffer =
- tree = #(nepodařilo(se, otevřít(Soubor)))
Nonprojective Parsing Example

- stack = #.
- buffer =
- tree = #(nepodařilo(se, otevřít(Soubor)))

RARC

- stack = #
- buffer =
- tree = #(nepodařilo(se, otevřít(Soubor)),.)
Malt and MST Accuracy

• Czech (PDT):
  – MST Parser over 85%
  – Malt Parser over 86%
    • Sentence accuracy (“complete match”) 35%, that is high!
  – The two parsers use different strategies and can be combined (either by voting (third parser needed) or one preparing features for the other)

• Other languages (CoNLL shared tasks)
  – MST was slightly better on most languages.
  – Accuracies not comparable cross-linguistically, figures are very dependent on particular corpora.
Features Are the Key to Success

• Common feature of MST and Malt:
  – Both can use large number of input text features.
  – Nontrivial machine learning algorithm makes sure that the important features will be given higher weight.
  – Machine learning algorithms are general classifiers.
    • Typically there is a library ready to download.
    • The concrete problem (here tree building) must be converted to a sequence of classification decisions, e.g. vectors (feature values + answer).