

Language as a Tree

Mariana Romanyshyn
Grammarly, Inc.

Contents

1. Syntactic trees in use
2. Brief overview of constituency parsing
3. Dependency parsing
 - a. algorithms
 - b. metrics
4. Parsing errors



1. Syntactic trees in use

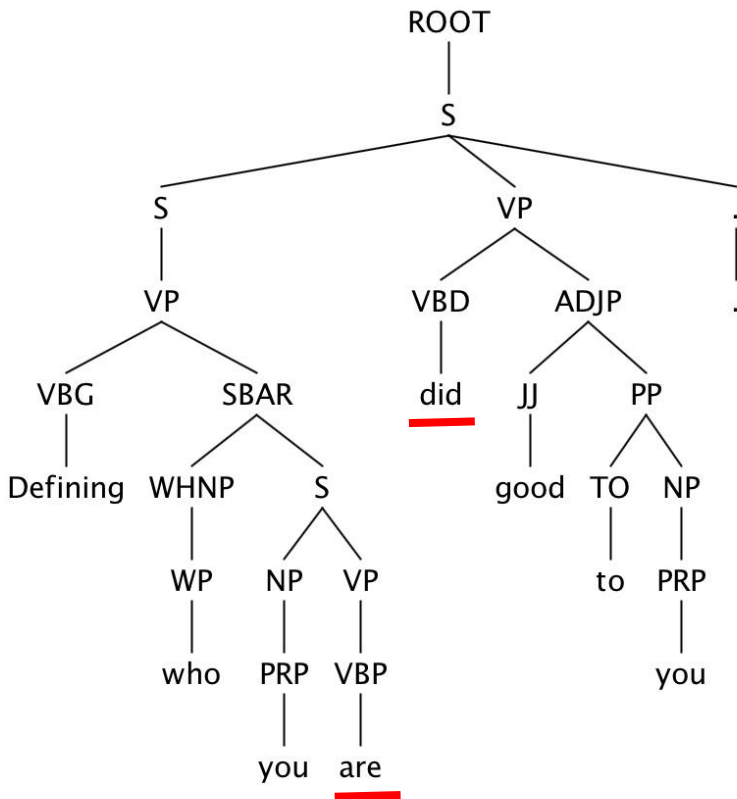
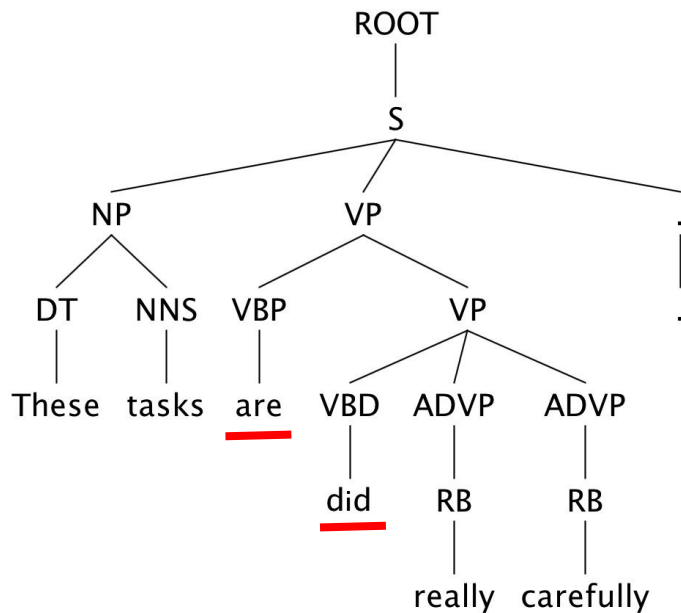
Error correction

These tasks are did really carefully.

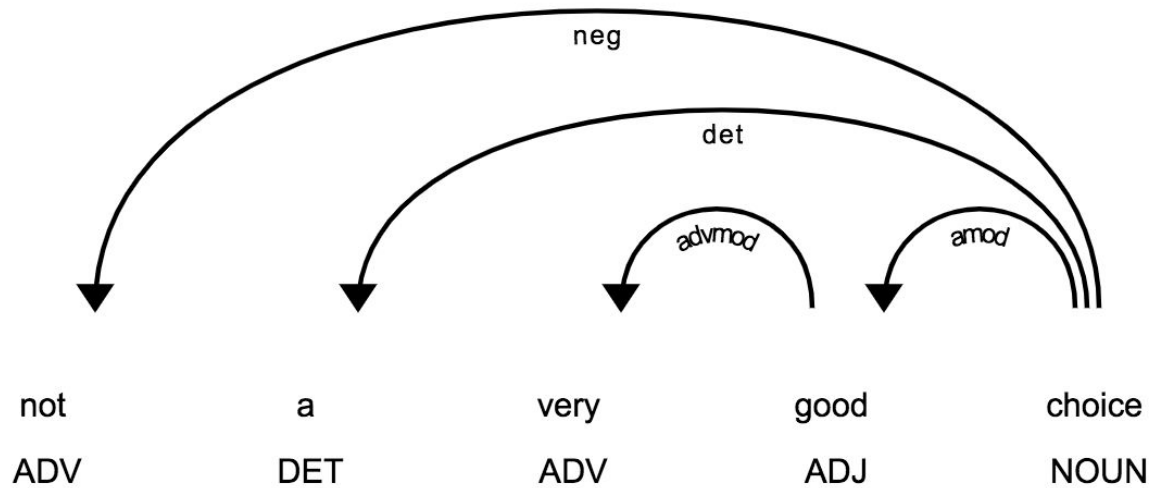
~~are did~~ → are done

Defining who you are did good to you.

Error correction



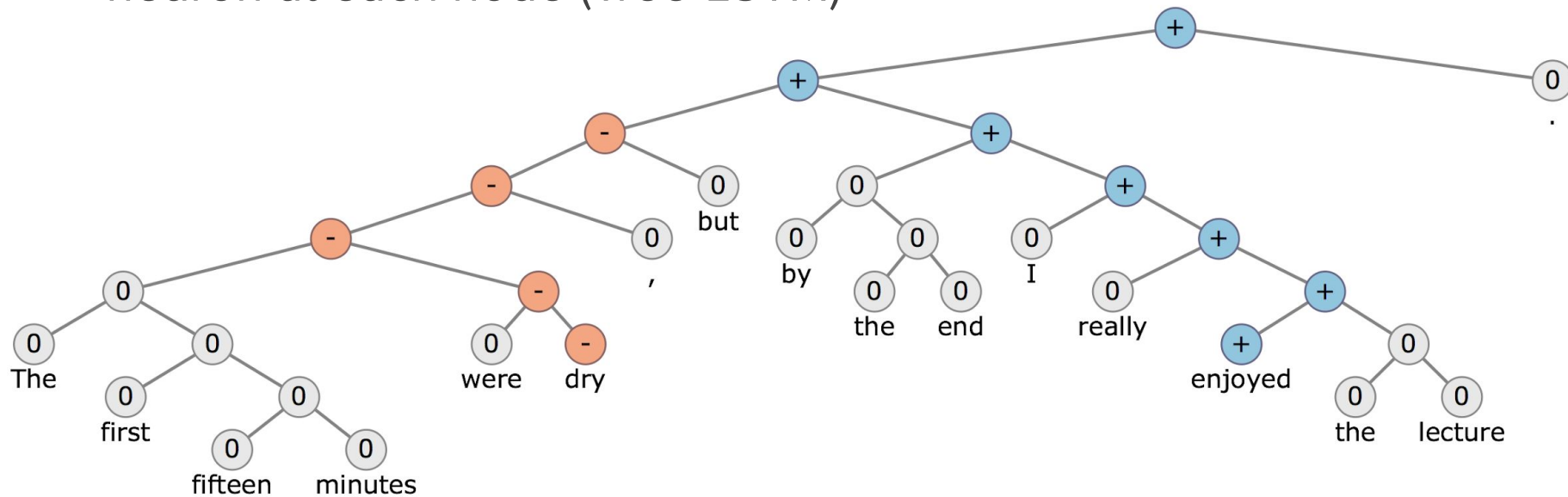
Sentiment Analysis



Negation spans all children of the parent.

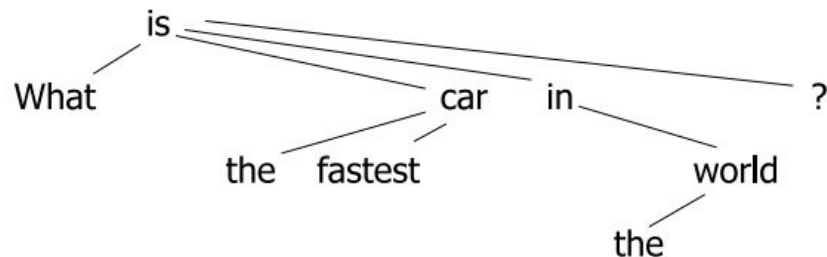
Sentiment Analysis

- rules at each node
- classifier at each node
- neuron at each node (Tree-LSTM)

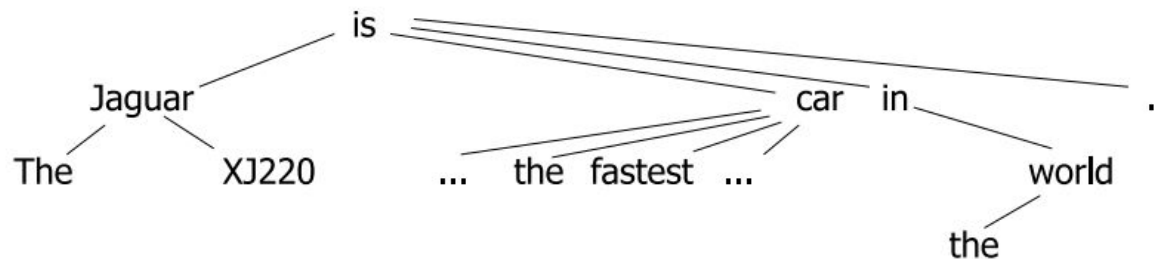


Question Answering

What is the fastest car in the world ?



The Jaguar XJ220 is the dearest, fastest and the most sought after car in the world .



Fact Extraction

Bloomberg ▼

Cantor Fitzgerald Sued by Partners Who Moved to Reorient

China Lawsuit

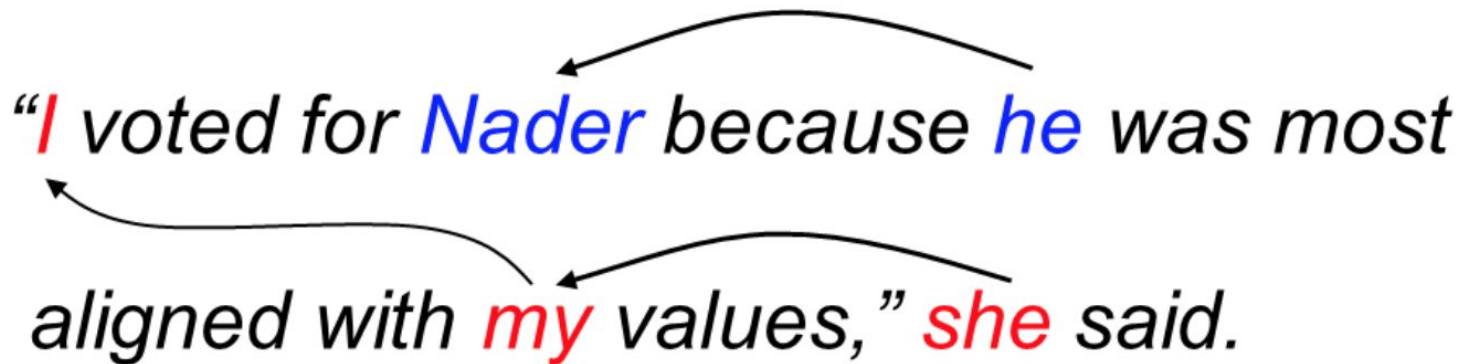
In 2011, Cantor filed a lawsuit in China against Boyer, Ainslie and other traders who left its Hong Kong office, accusing them of breaching their employment agreements and causing a 29 percent drop in average monthly revenue at the branch. Two years later, Cantor officials settled their claims against the former executives, according to filings with the Hong Kong Stock Exchange. The terms weren't made public.

Sheryl Lee, a Cantor spokeswoman, said today by phone that the company has a policy of not commenting on litigation.

Coreference Resolution

Needed for

- entity linking
- text summarization
- question answering
- fact extraction...

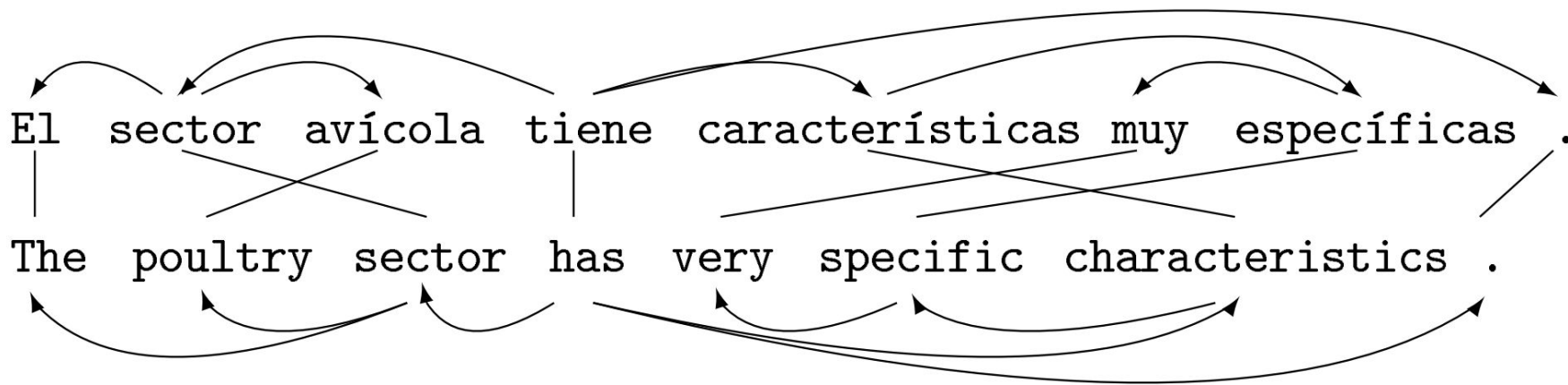


"I voted for Nader because he was most aligned with my values," she said.

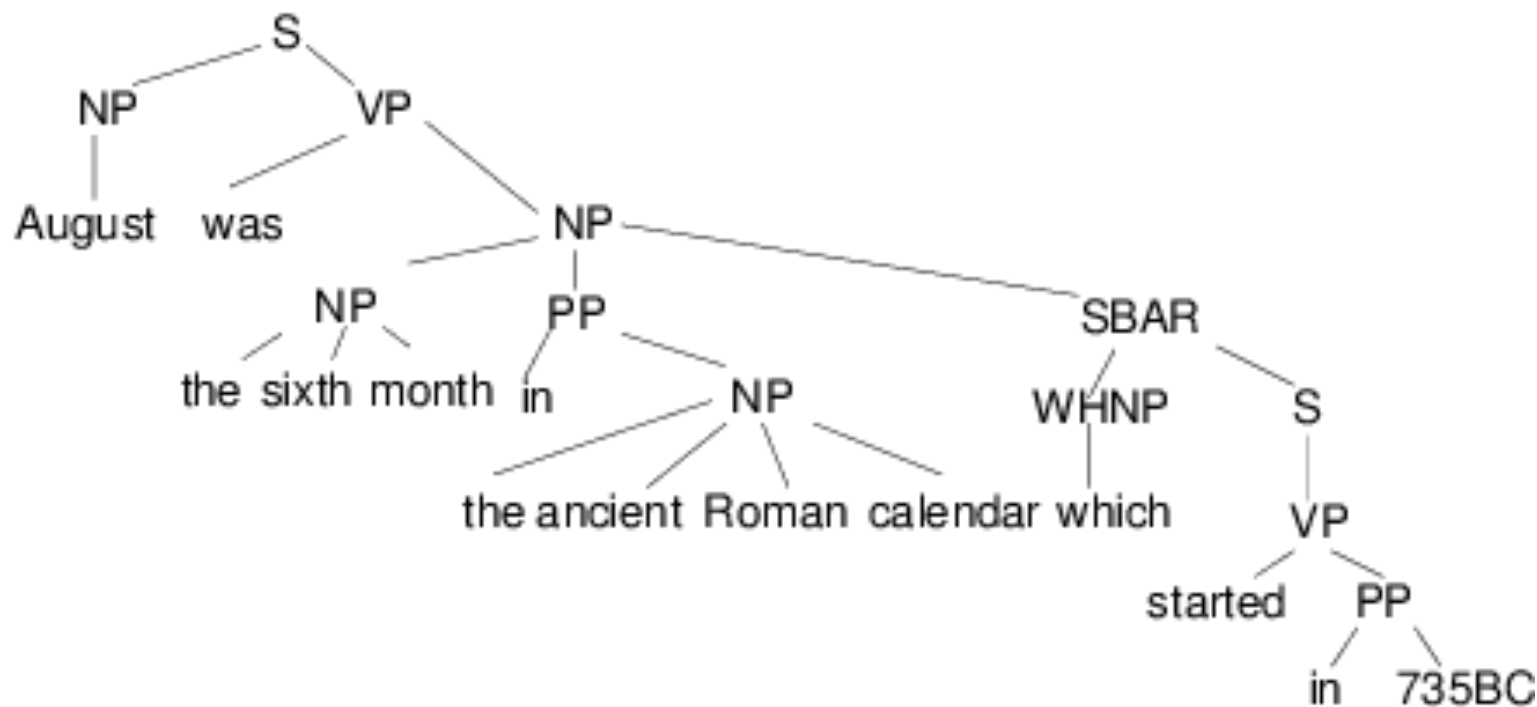
The diagram shows three curved arrows indicating coreference relations: one from 'I' to 'she', one from 'he' to 'Nader', and one from 'my' to 'she'.

Machine Translation

- parallel treebanks
- tree alignment models for reordering words
- syntactic language models for reranking



Text Simplification

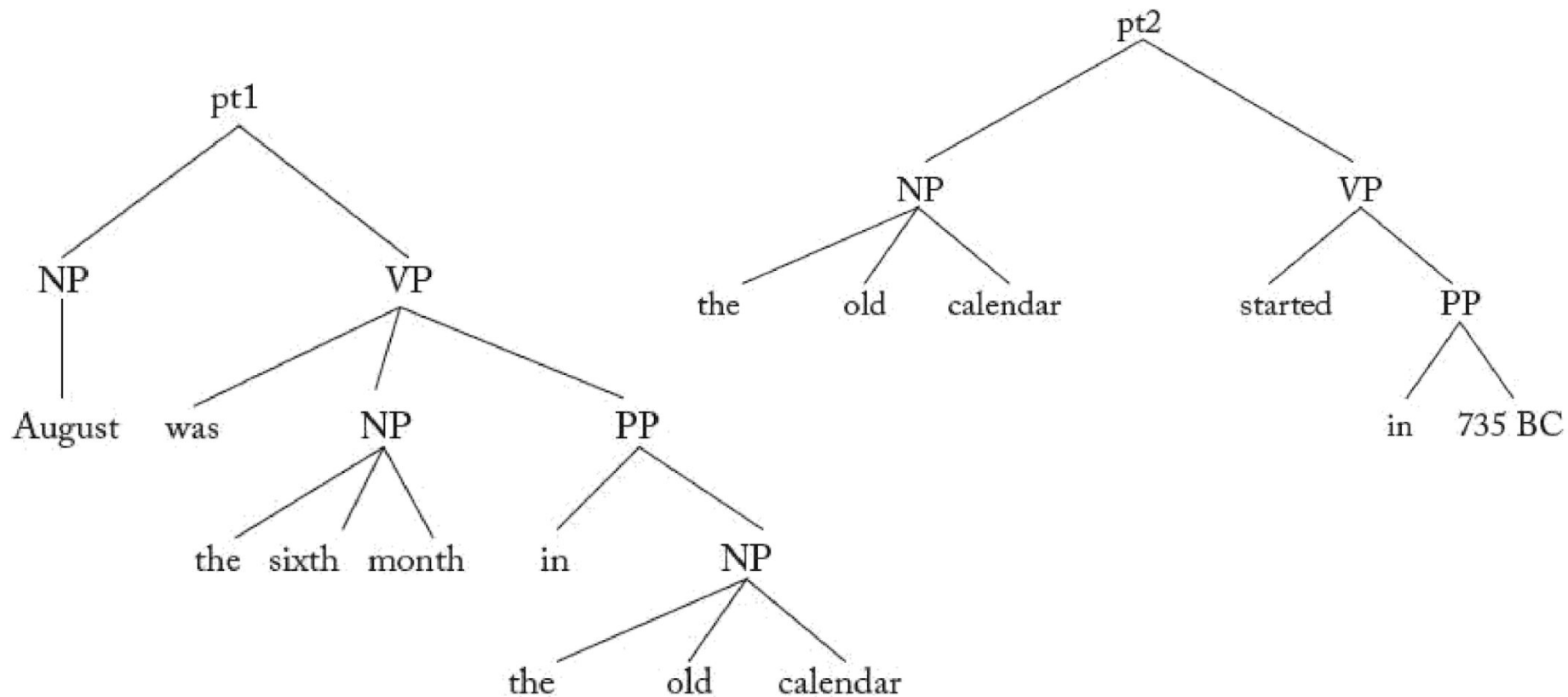


Text Simplification

Operations on the parse tree:

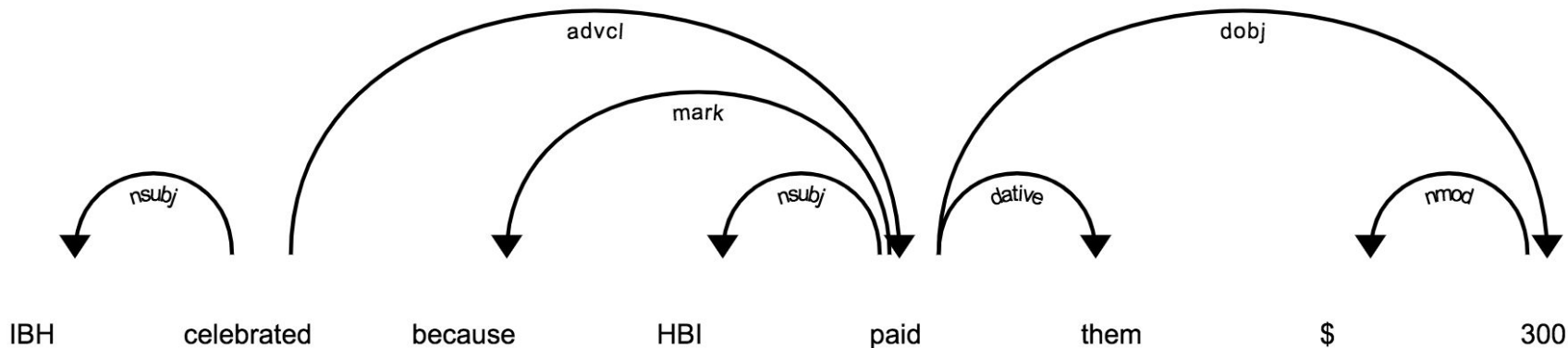
- split
- drop
- reorder
- substitute

Text Simplification



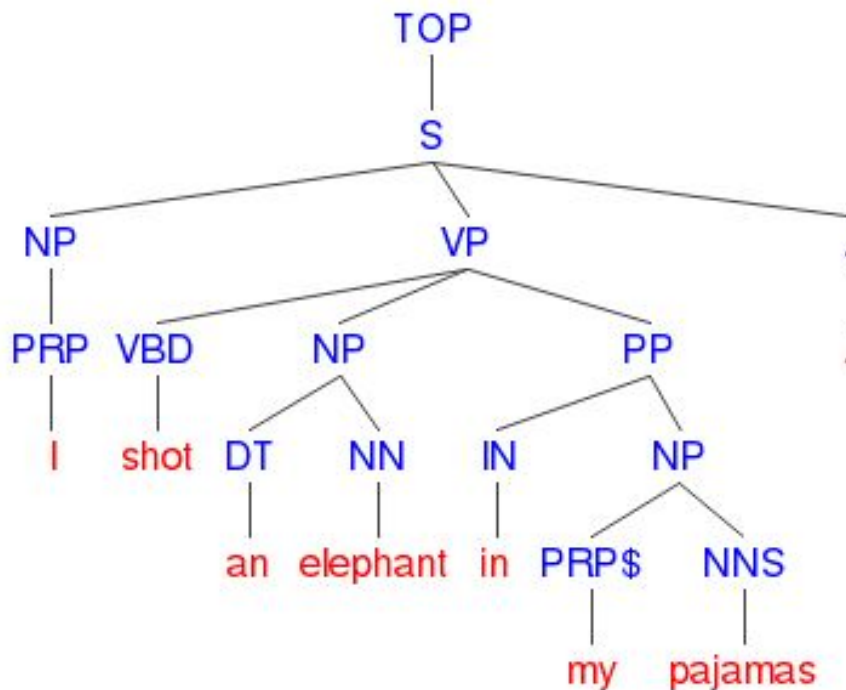
Features

- dependency label
- parent features
- paths to NEs
 - *HBI: dative_nsubj*
 - *IBH: dative_advcl_nsubj*
- path to the root: *dative_advcl_root*
- depth in the tree...



Features

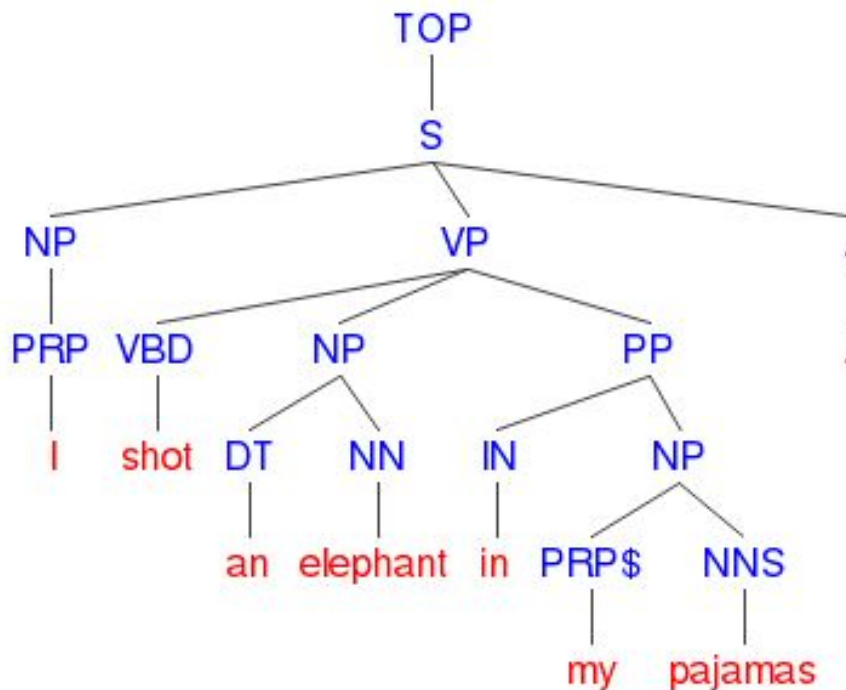
- constituency label
- head node features
- closest common parent
- spans
- path to the root
 - *NP_VP_S_TOP*
- paths to other elements
 - *NP<-VP->PP->NP*
 - *NP<-VP<-S->NP*
- depth in the tree...



2. Brief overview of constituency parsing

Constituency parsing

- appeared in 1900s, was formalized in 1950s
- breaks a sentence into independent constituents
- operates at the phrase level



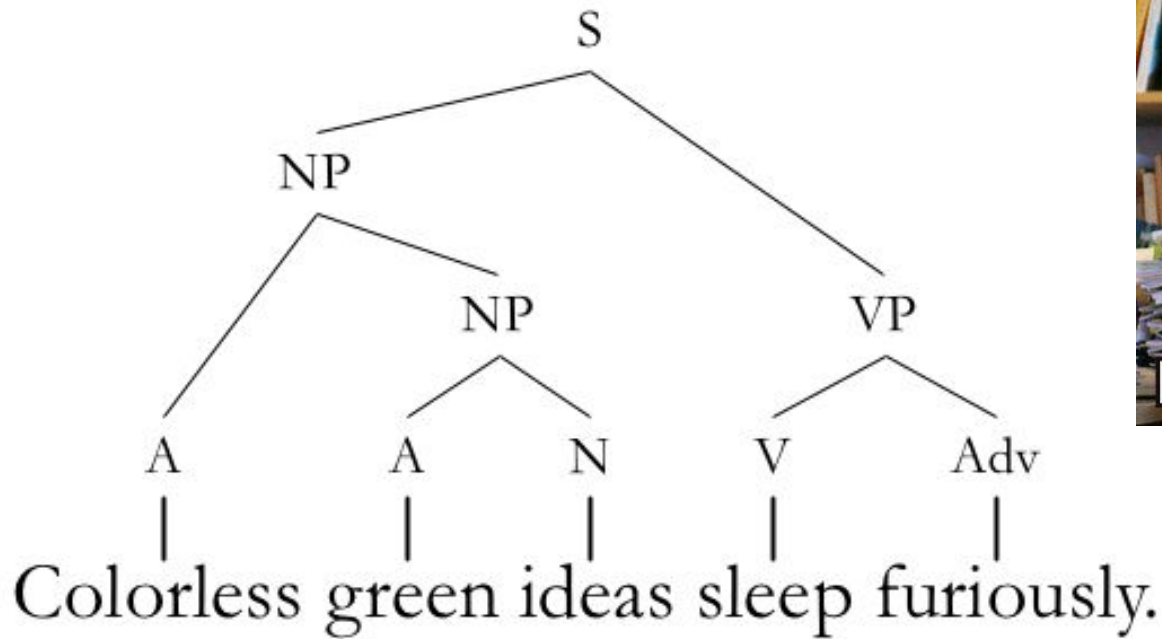
Constituency trees - bracketed format

```
(TOP (S (PP (IN With)
           (NP (NP (NNS celebrations))
                (PP (IN for)
                    (NP (NP (DT the)
                        (JJ long-anticipated)
                        (NN start))
                    (PP (IN of)
                        (NP (DT the) (NN year) (CD 2000))))))
                (ADVP (RB barely) (RB over))))
      (, ,)
      (NP-TMP (NN today))
      (NP-SBJ-1 (JJ Chinese)
                (NNS people))
      (VP (VBD began)
          (ADVP (RB busily))
          (VP (VBG preparing)
              (S (NP-SBJ (-NONE- *PRO*-1))
                  (VP (TO to)
                      (VP (VB mark)
                          (NP (DT another) (JJ new) (NN year))))))))
      (. .)))
```

Treebanks

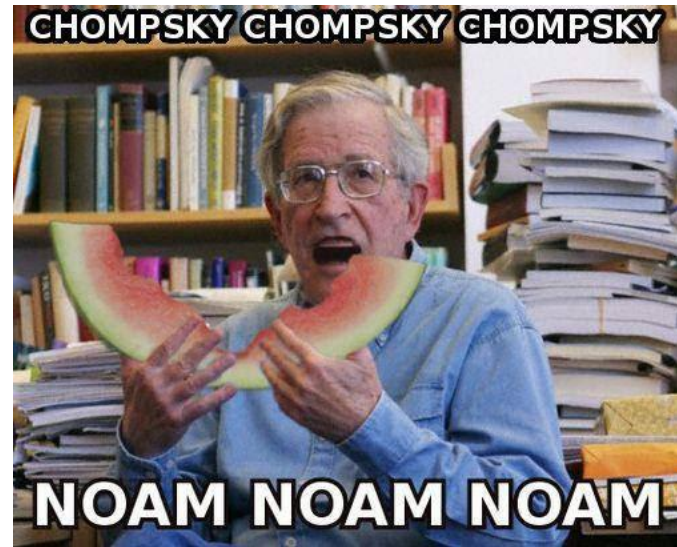
- Usage:
 - Testing linguistic hypotheses
 - Training and evaluation of syntactic parsers
- Problems:
 - Costly
 - May contain errors
 - May use different notations





VS.

Furiously sleep ideas green colorless.



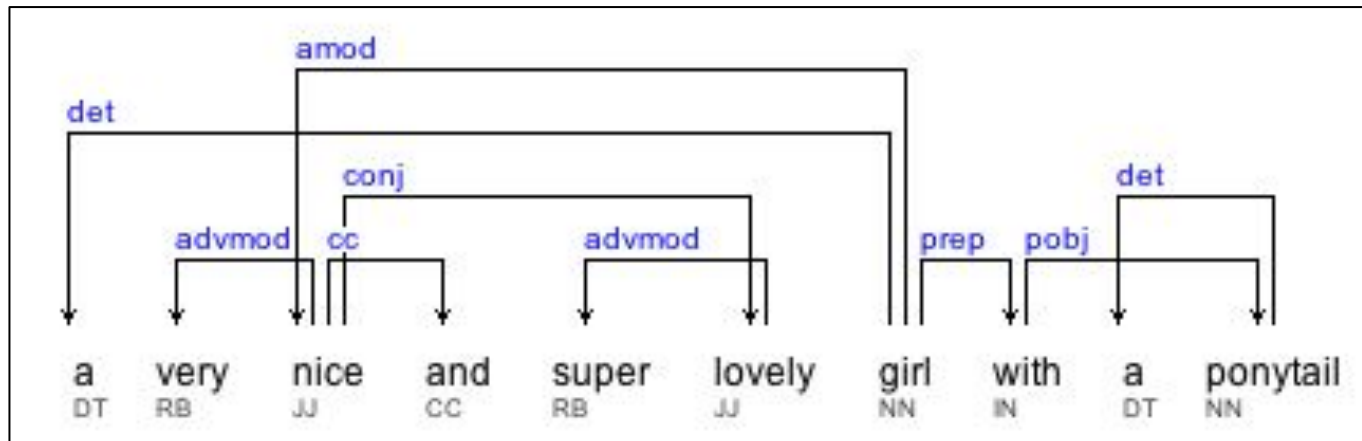
Constituency parsing algorithms

- Top-down
 - start from ROOT and try to match input sentence
- Bottom-up
 - start from input sentence and try to match ROOT
- Dynamic programming
 - try all combinations and store partial results on the way

3. Dependency parsing

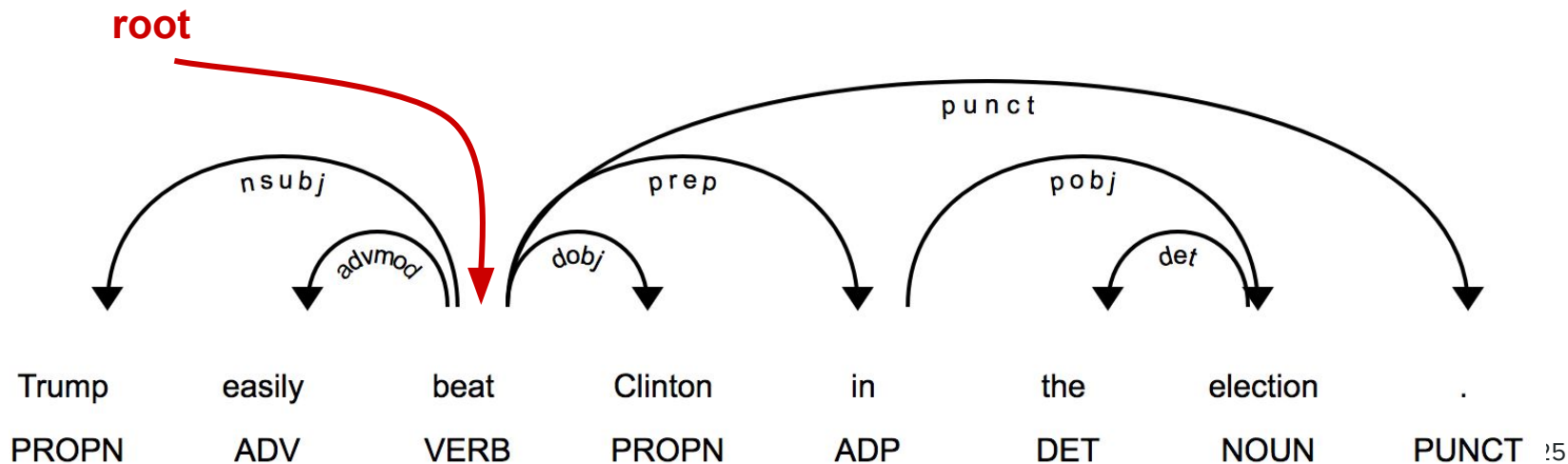
Dependency parsing

- appeared in 2000s
- represents the relations between the words in the sentence
- operates at the word level
- good solution for more synthetic languages



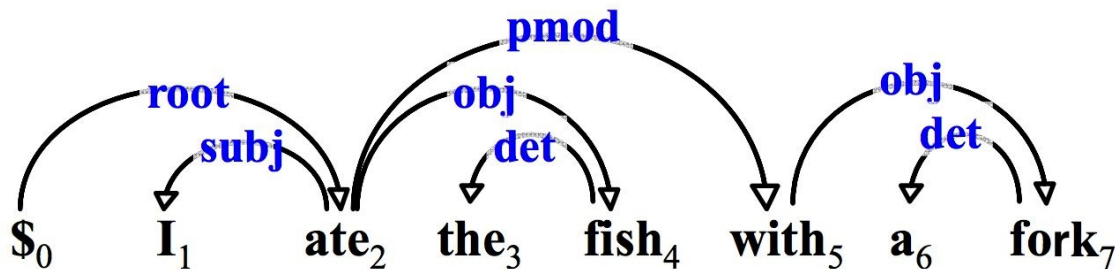
Dependency parsing

- every **child** has exactly one **parent**
- dependencies must form a tree
- the tree ends with a **root** node

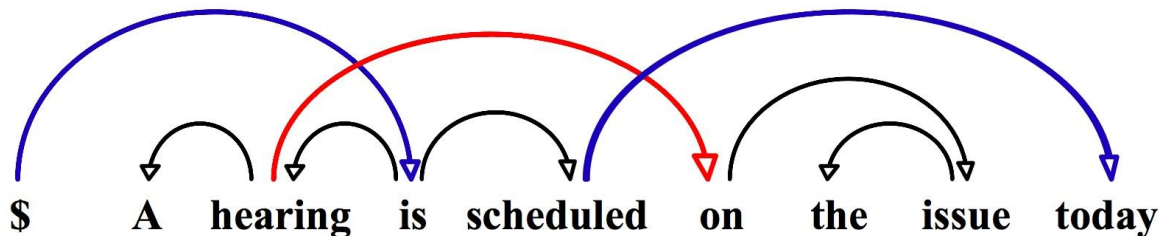


Projectivity

- Projective tree

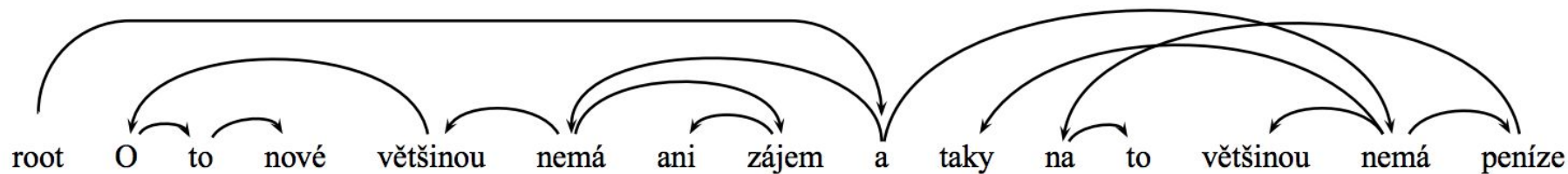
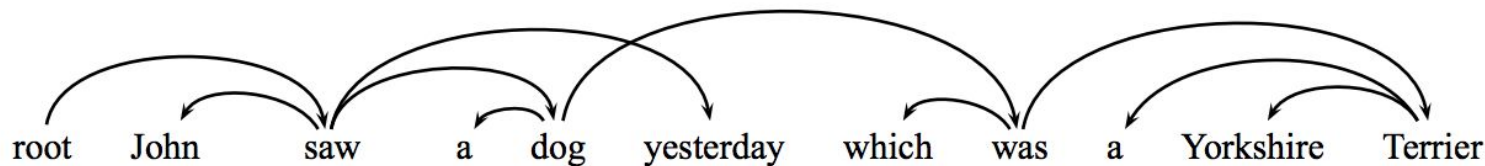


- Non-projective tree



Projectivity

- Non-projective trees in English and Czech



He is mostly not even interested in the new things and in most cases, he has no money for it either.

Dependency treebanks

- Converted from constituency trees
- Prague Dependency Treebank for Czech
- Universal Dependencies Treebank
 - more than 100 treebanks
 - over 70 languages

Universal Dependency Treebank

1	If	if	IN	3	mark
2	you	you	PRP	3	nsubj
3	want	want	VBP	14	advcl
4	to	to	TO	5	aux
5	receive	receive	VB	3	xcomp
6	e-mails	e-mail	NNS	5	dobj
7	about	about	IN	6	prep
8	my	my	PRP\$	10	poss
9	upcoming	upcoming	JJ	10	amod
10	shows	show	NNS	7	pobj
11	,	,	,	14	punct
12	then	then	RB	14	advmod
13	please	please	UH	14	intj
14	give	give	VB	0	root
15	me	me	PRP	14	dative
...					

Graph-based dependency parsing

Idea:

- find the highest score tree from a complete graph.

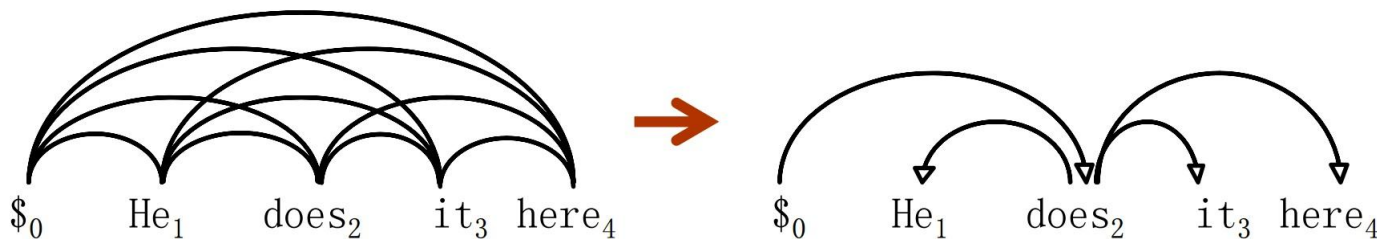
Pros:

- performs better on long-distance dependencies
- allows non-projective trees

Cons:

- slow

Graph-based dependency parsing



$$Y^* = \arg \max_{Y \in \Phi(X)} score(X, Y)$$

$$score(X, Y) = \sum_{(h, m) \in Y} score(X, h, m)$$

X – sentence

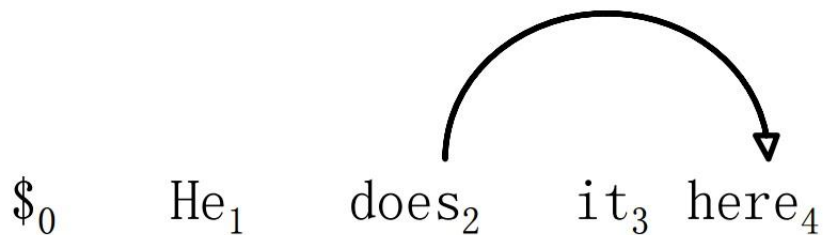
Y – candidate tree

h – head

m – modifier

Features

$$\text{score}(2,4) = ?$$



Each link is a feature vector: **$\text{score}(2, 4) = w * f(2,4)$**

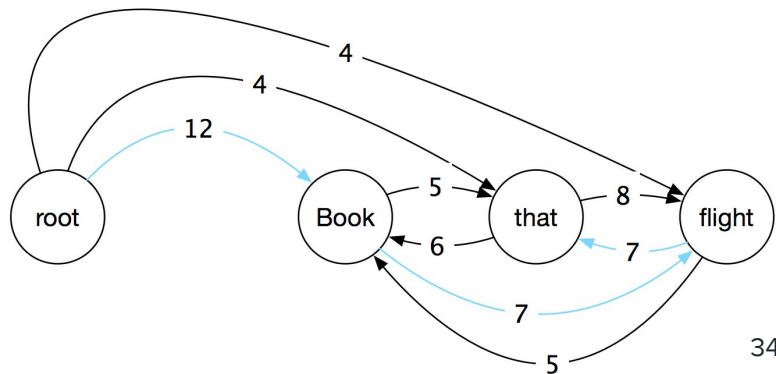
Example from slides of Rush and Petrov (2012)

* As McGwire neared , fans went wild

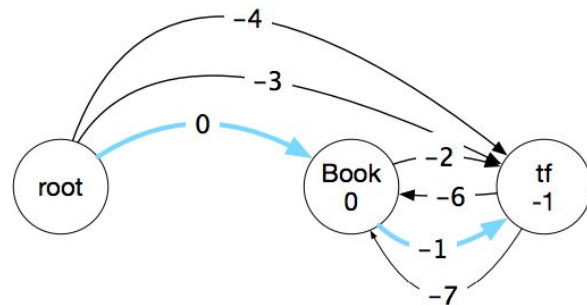
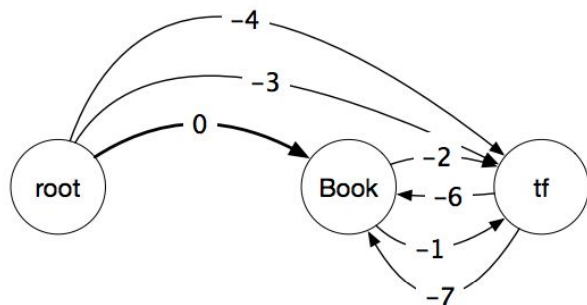
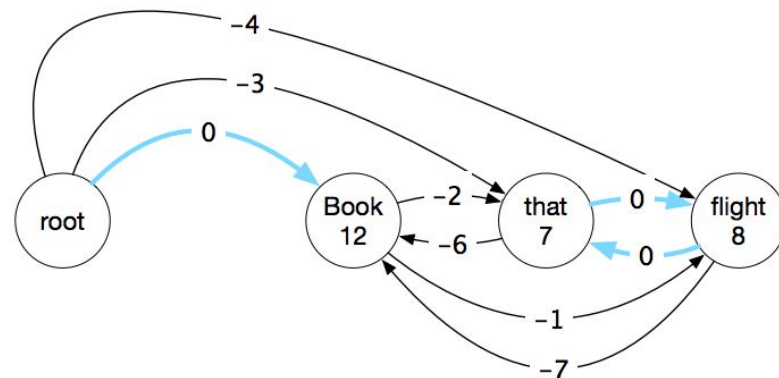
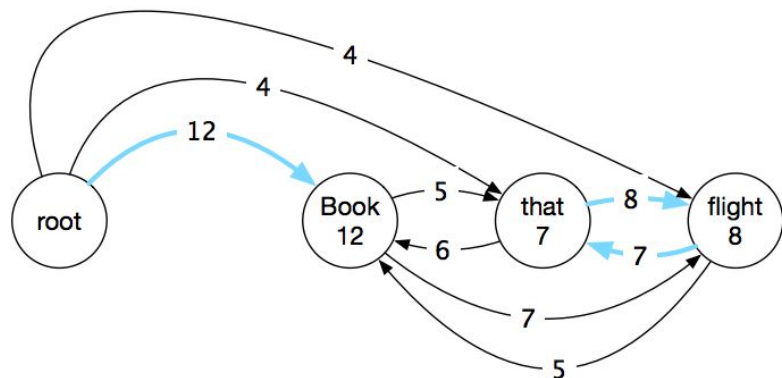
[went]	[VBD]	[As]	[ADP]	[went]
[VERB]	[As]	[IN]	[went, VBD]	[As, ADP]
[went, As]	[VBD, ADP]	[went, VERB]	[As, IN]	[went, As]
[VERB, IN]	[VBD, As, ADP]	[went, As, ADP]	[went, VBD, ADP]	[went, VBD, As]
[ADJ, *, ADP]	[VBD, *, ADP]	[VBD, ADJ, ADP]	[VBD, ADJ, *]	[NNS, *, ADP]
[NNS, VBD, ADP]	[NNS, VBD, *]	[ADJ, ADP, NNP]	[VBD, ADP, NNP]	[VBD, ADJ, NNP]
[NNS, ADP, NNP]	[NNS, VBD, NNP]	[went, left, 5]	[VBD, left, 5]	[As, left, 5]
[ADP, left, 5]	[VERB, As, IN]	[went, As, IN]	[went, VERB, IN]	[went, VERB, As]
[JJ, *, IN]	[VERB, *, IN]	[VERB, JJ, IN]	[VERB, JJ, *]	[NOUN, *, IN]
[NOUN, VERB, IN]	[NOUN, VERB, *]	[JJ, IN, NOUN]	[VERB, IN, NOUN]	[VERB, JJ, NOUN]
[NOUN, IN, NOUN]	[NOUN, VERB, NOUN]	[went, left, 5]	[VERB, left, 5]	[As, left, 5]
[IN, left, 5]	[went, VBD, As, ADP]	[VBD, ADJ, *, ADP]	[NNS, VBD, *, ADP]	[VBD, ADJ, ADP, NNP]
[NNS, VBD, ADP, NNP]	[went, VBD, left, 5]	[As, ADP, left, 5]	[went, As, left, 5]	[VBD, ADP, left, 5]
[went, VERB, As, IN]	[VERB, JJ, *, IN]	[NOUN, VERB, *, IN]	[VERB, JJ, IN, NOUN]	[NOUN, VERB, IN, NOUN]
[went, VERB, left, 5]	[As, IN, left, 5]	[went, As, left, 5]	[VERB, IN, left, 5]	[VBD, As, ADP, left, 5]
[went, As, ADP, left, 5]	[went, VBD, ADP, left, 5]	[went, VBD, As, left, 5]	[ADJ, *, ADP, left, 5]	[VBD, *, ADP, left, 5]
[VBD, ADJ, ADP, left, 5]	[VBD, ADJ, *, left, 5]	[NNS, *, ADP, left, 5]	[NNS, VBD, ADP, left, 5]	[NNS, VBD, *, left, 5]
[ADJ, ADP, NNP, left, 5]	[VBD, ADP, NNP, left, 5]	[VBD, ADJ, NNP, left, 5]	[NNS, ADP, NNP, left, 5]	[NNS, VBD, NNP, left, 5]
[VERB, As, IN, left, 5]	[went, As, IN, left, 5]	[went, VERB, IN, left, 5]	[went, VERB, As, left, 5]	[JJ, *, IN, left, 5]
[VERB, *, IN, left, 5]	[VERB, JJ, IN, left, 5]	[VERB, JJ, *, left, 5]	[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

Graph-based dependency parsing

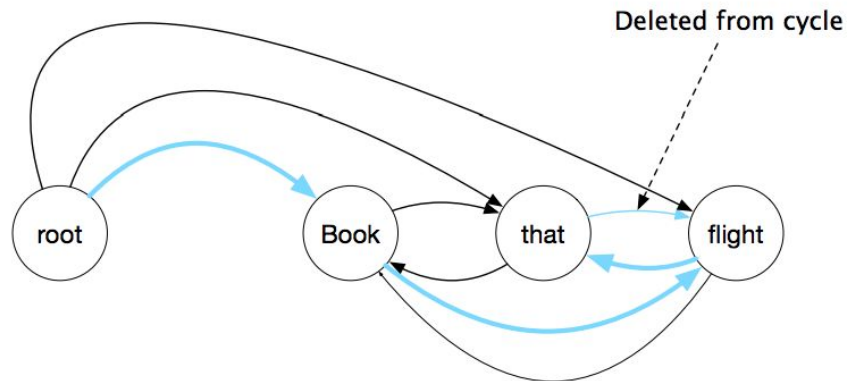
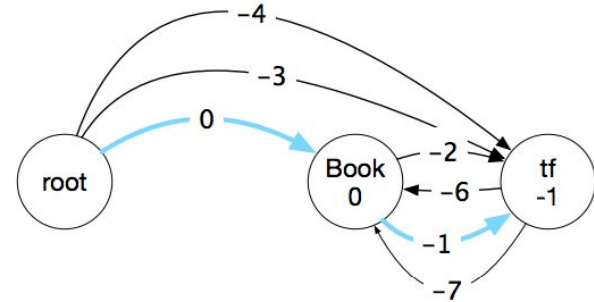
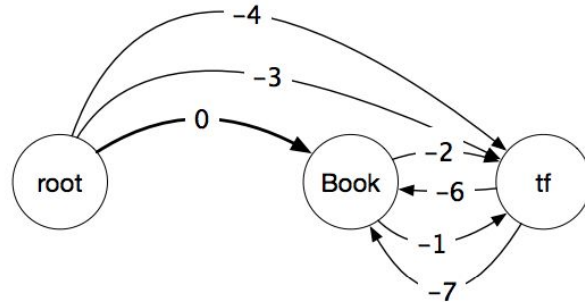
- Maximum directed spanning tree (MST)
 - trace edges with maximum score
 - if a cycle appears (recursively):
 - adjust scores - subtract max incoming score from all incoming scores of each node
 - collapse cycling nodes
 - apply MST to new graph
 - clean up



Graph-based dependency parsing



Graph-based dependency parsing



Transition-based dependency parsing

Idea:

- apply transition actions one by one from left to right

Pros:

- fast

Cons:

- performs worse on long-distance dependencies
- allows only projective trees

Transition-based parsing

Configurations:

- ***queue*** - the words of the sentence
- ***stack*** - words yet without head
- ***set of relations***

Transition-based parsing (Arc-Eager)

Actions:

- ***shift*** - move the word from the queue onto the stack
- ***right-arc*** - create a right dependency arc between the word on top of the stack and the next token in the queue
- ***left-arc*** - create a left dependency arc between the word on top of the stack and the next token in the queue
- ***reduce*** - pop the stack, removing only its top item, as long as that item has a head

Transition-based parsing

$$\begin{aligned} Y^* &= \arg \max_{Y \in \Phi(X)} \text{score}(X, Y) \\ &= \arg \max_{a_0 \dots a_m \rightarrow Y} \sum_{i=0}^m \text{score}(X, h_i, a_i) \end{aligned}$$

X – sentence

Y – candidate tree

a – action

h – partial result built so far

m – number of words in the sentence (== number of actions)

Transition-based parsing

Now:

- do a sequence of actions through the space of possible configurations
- apply an action to a configuration and produce a new configuration

function DEPENDENCYPARSE(*words*) **returns** dependency tree

state \leftarrow {[root], [*words*], [] } ; initial configuration

while *state* **not** final

t \leftarrow ORACLE(*state*) ; choose a transition operator to apply

 state \leftarrow APPLY(*t*, *state*) ; apply it, creating a new state

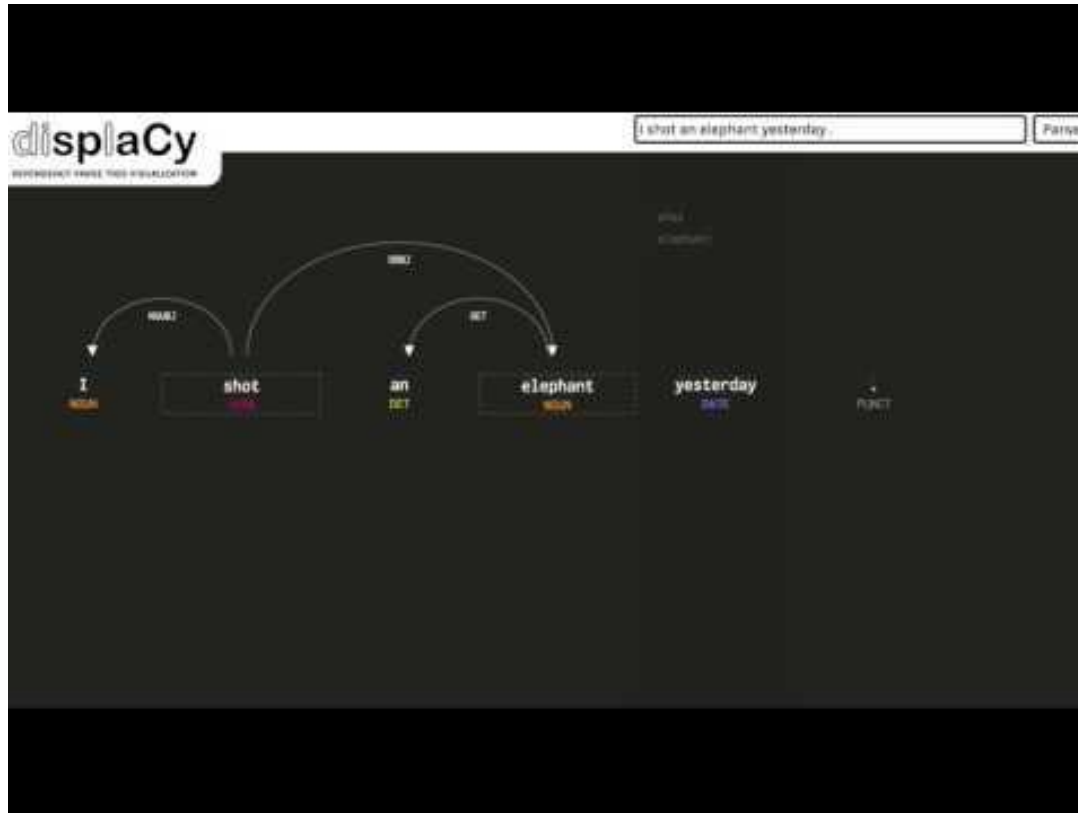
return *state*

Transition-based parsing

Build a parse tree for the sentence below:

A large elephant was wearing my pyjamas

Transition-based parsing: demo



Transition-based parsing for non-projective trees

Actions:

- **shift** - move the word from the queue onto the stack
- **right-arc** - create a right dependency arc between the word on top of the stack and the next token in the queue
- **left-arc** - create a left dependency arc between the word on top of the stack and the next token in the queue
- **reduce** - pop the stack, removing only its top item, as long as that item has a head
- **swap** - exchange the words on top of the stack and on top of the queue

Training a transition-based parser

```
training_set ← []  
for sentence, tree pair in corpus do  
    sequence ← oracle(sentence, tree)  
    configuration ← initialize(sentence)  
    while not configuration.IsFinal() do  
        action ← sequence.next()  
        features ←  $\phi$ (configuration)  
        training_set.add(features, action)  
        configuration ← configuration.apply(action)  
  
train a classifier on training_set
```

Oracles

Oracle - a function that retrieves the transition at each point in tree.

- static oracle
 - checks: left/right arc => reduce => shift
 - returns the first satisfactory transition
- non-deterministic oracle
 - checks: left/right arc, reduce, shift
 - returns all ***valid*** transitions

Oracles

Oracle - a function that retrieves the transition at each point in tree.

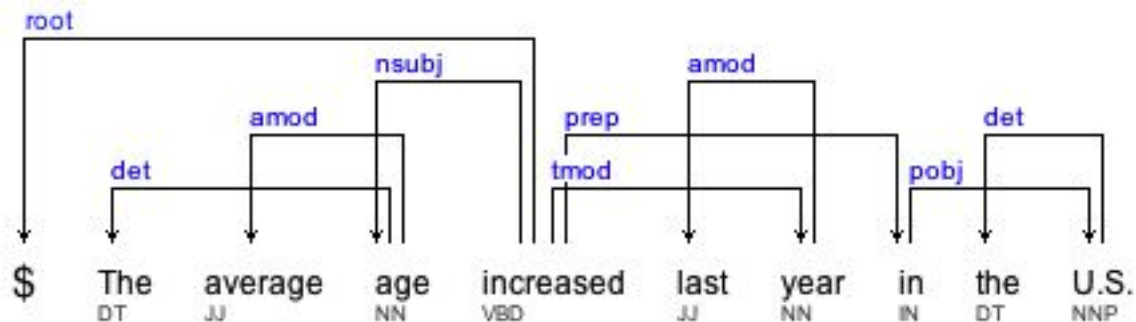
- dynamic
 - train a classifier to decide on the action
 - use golden tree for training
 - return transactions with the lowest loss

Dependency parsing metrics

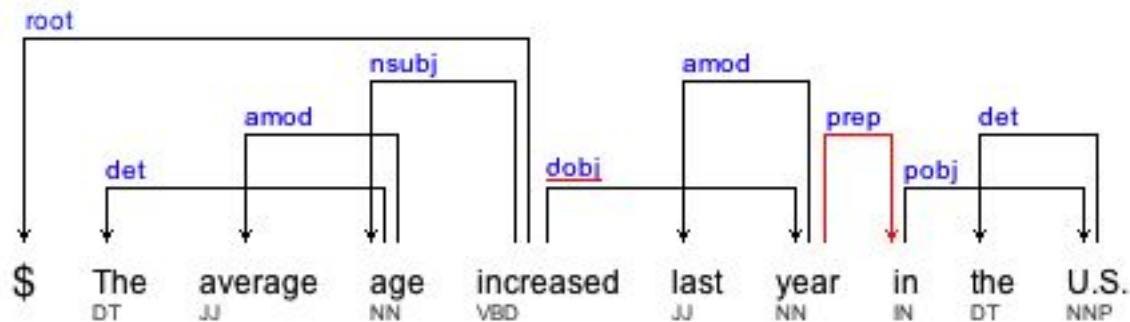
- Unlabeled Attachment Score
 - percentage of words that have correct heads
- Labeled Attachment Score
 - percentage of words that have correct heads and labels
- Recall/Precision/F-measure on separate labels
- Root Accuracy
- Complete Match

Dependency parsing metrics

- Gold tree



- Produced tree



Use case

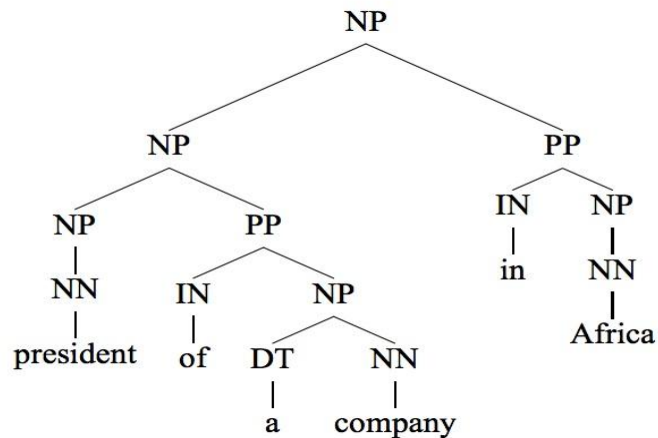
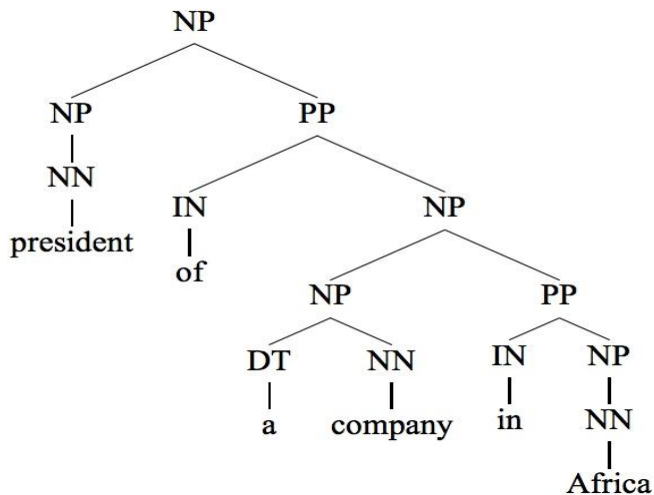
Transition-based arc-eager unlabeled dependency parser for Estonian

<https://github.com/mariana-scorp/esscass-2019-nlp/tree/master/3-tree/dep-parser-est.ipynb>

4. Parsing errors

The attachment problem

- PP attachment



The attachment problem

- PP attachment
- NP attachment
 - *We [decided to [build a museum this week]].*
 - *We [decided to [build a museum] this week].*

The attachment problem

- PP attachment
- NP attachment
- Modifier attachment
 - *[[old women] and men]*
 - *[old [women and men]]*

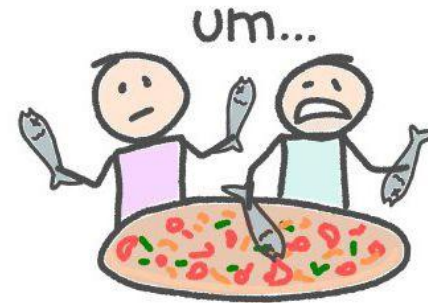
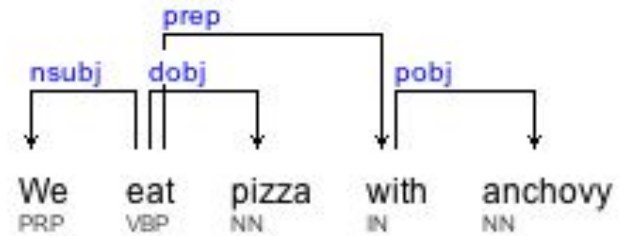
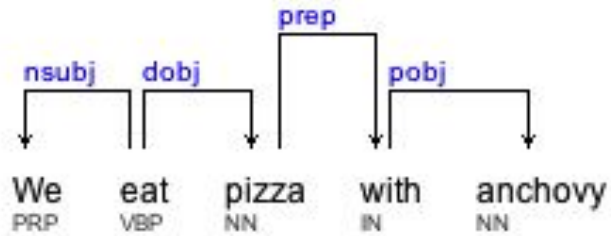
The attachment problem

- PP attachment
- NP attachment
- Modifier attachment
- Clause attachment
 - *[[I'm glad I'm a man], and [so is Lola]].*
 - *[I'm glad [[I'm a man], and [so is Lola]]].*

The attachment problem

- PP attachment
- NP attachment
- Modifier attachment
- Clause attachment
- VP attachment
 - *We have [to pay Tom [[to do the job] and [to manage everything]]].*
 - *We have [[to pay Tom [to do the job]] and [to manage everything]].*

The attachment problem



The PP attachment problem: solutions

- Majority class (noun attachment) wins
- Most likely class for each preposition wins
- Binary classification using maximum likelihood estimation

1. **If** $f(v, n1, p, n2) > 0$

$$\hat{p}(1|v, n1, p, n2) = \frac{f(1, v, n1, p, n2)}{f(v, n1, p, n2)}$$

2. **Else if** $f(v, n1, p) + f(v, p, n2) + f(n1, p, n2) > 0$

$$\hat{p}(1|v, n1, p, n2) = \frac{f(1, v, n1, p) + f(1, v, p, n2) + f(1, n1, p, n2)}{f(v, n1, p) + f(v, p, n2) + f(n1, p, n2)}$$

3. **Else if** $f(v, p) + f(n1, p) + f(p, n2) > 0$

$$\hat{p}(1|v, n1, p, n2) = \frac{f(1, v, p) + f(1, n1, p) + f(1, p, n2)}{f(v, p) + f(n1, p) + f(p, n2)}$$

4. **Else if** $f(p) > 0$

$$\hat{p}(1|v, n1, p, n2) = \frac{f(1, p)}{f(p)}$$

5. **Else** $\hat{p}(1|v, n1, p, n2) = 1.0$ (default is noun attachment).

The PP attachment problem: solutions

- Majority class (noun attachment) wins
- Most likely class for each preposition wins
- Binary classification using maximum likelihood estimation:
 - $P(\text{eat, pizza, with, anchovy})$
 - $P(\text{eat, pizza, with}), P(\text{eat, with, anchovy}), P(\text{pizza, with, anchovy})$
 - $P(\text{eat, with}), P(\text{with, anchovy}), P(\text{pizza, with})$
 - $P(\text{with})$

The coordination attachment problem: solutions


- The closer relation wins
- Similarity of head nodes in coordination
 - books about musical instruments and other literature
 - dogs in houses and cats
 - cats with fleas and dogs
 - men who like shopping and women

More things to improve

- Fixing part-of-speech errors while building trees
- Exploring richer features
 - *e.g., mark coordination, grandparents, siblings*
- Reranking of n-best parse trees
 - *lexicalization, ancestors, functional/lexical heads*
 - *tree ngrams, rightmost-branch bias*
 - *coordination parallelism*
- Ensembles of parsers
- Semi-supervised learning
- Beam search



Thank you !



Any questions ?

References

- [Speech and Language Processing](#), Chapters 10-13, Jurafsky and Martin (2018)
- [Syntax and Parsing](#), Yoav Goldberg, (2017)
- [Prepositional Phrase Attachment through a Backed-Off Model](#), Michael Collins and James Brooks (1995)
- [Accurate Unlexicalized Parsing](#), Dan Klein and Chris Manning (2003)
- [Non-projective Dependency Parsing using Spanning Tree Algorithms](#), Ryan McDonald et al. (2005)
- [Improvements in Transition Based Systems for Dependency Parsing](#), Francesco Sartorio (2015)
- [Parsing English in 500 Lines of Python](#), Matthew Honnibal (2013)
- [The Dirty Little Secret of Constituency Parser Evaluation](#), Romanysyn and Dyomkin (2014)