Dependency Parsing

- The problem:
  - Input: Sentence $x=w_0, w_1, \ldots, w_n$ with w_0=root
  - Output: Dependency graph $G = (V, A)$ for x where:
    - $V=\{0, 1, \ldots, n\}$ is the vertex set,
    - A is the arc set, i.e., $(i, j, k) \in A$ represents a dependency from w_i to w_j with label $l_k \in L$

Dependency Parsing

- Two main approaches:
  - Grammar-based parsing
    - Context-free dependency grammar
    - Constraint dependency grammar
  - Data-driven parsing
    - Transition-based models
    - Graph-based models

Grammar-based Parsing

- Context-free dependency grammar
  - Dependency grammar as lexicalized CFG:
    - $H \longrightarrow L_1 \cdot\cdot\cdot L_m\ h\ R_1 \cdot\cdot\cdot R_n$
    - $H \in V_N$; $h \in V_T$; $L_1 \cdot\cdot\cdot L_m, R_1 \cdot\cdot\cdot R_n \in V_N^*$
  - Standard context-free parsing algorithms
- Constraint dependency grammar
  - Parsing as constraint satisfaction:
    - Grammar consists of a set of boolean constraints, i.e. logical formulas that describe well-formed dependency graphs.
    - Constraint propagation removes candidate graphs that contradict constraints (eliminative parsing).

Data-driven Parsing

- Transition-based models
  - Define a transition system (state machine) for mapping a sentence to its dependency graph.
  - Learning: Induce a model for predicting the next state transition, given the transition history.
  - Parsing: Construct the optimal transition sequence, given the induced model.

Data-driven Parsing

- Graph-based models
  - Define a space of candidate dependency graphs for a sentence.
  - Learning: Induce a model for scoring an entire dependency graph for a sentence.
  - Parsing: Find the highest-scoring dependency graph, given the induced model.

Pros and Cons of Dependency Parsing

- Four types of considerations:
  - Complexity: faster than constituency
  - Transparency: direct encoding of predicate-argument structure
  - Word order: suitable for free word order languages
  - Expressivity: less expressive than constituency

Dependency Parsing

- Increasing interest, starting from the shared tasks on multilingual dependency parsing at CoNLL 2006 & 2007 and ending with UD
- Suitable to deal with languages with relatively free word order
- Influenced phrase structure parsing too (role of heads, bilexical relations for disambiguation, …)

CoNLL Shared Tasks

- CoNLL 2006 – Multilingual dependency parsing: Arabic, Bulgarian, Chinese, Czech, Danish, Dutch, German, English, Japanese, Polish, Slovene, Spanish, Swedish, Turkish
- CoNLL 2007:
  - Multilingual track: Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, Turkish
  - Domain Adaptation track

Stanford Dependencies

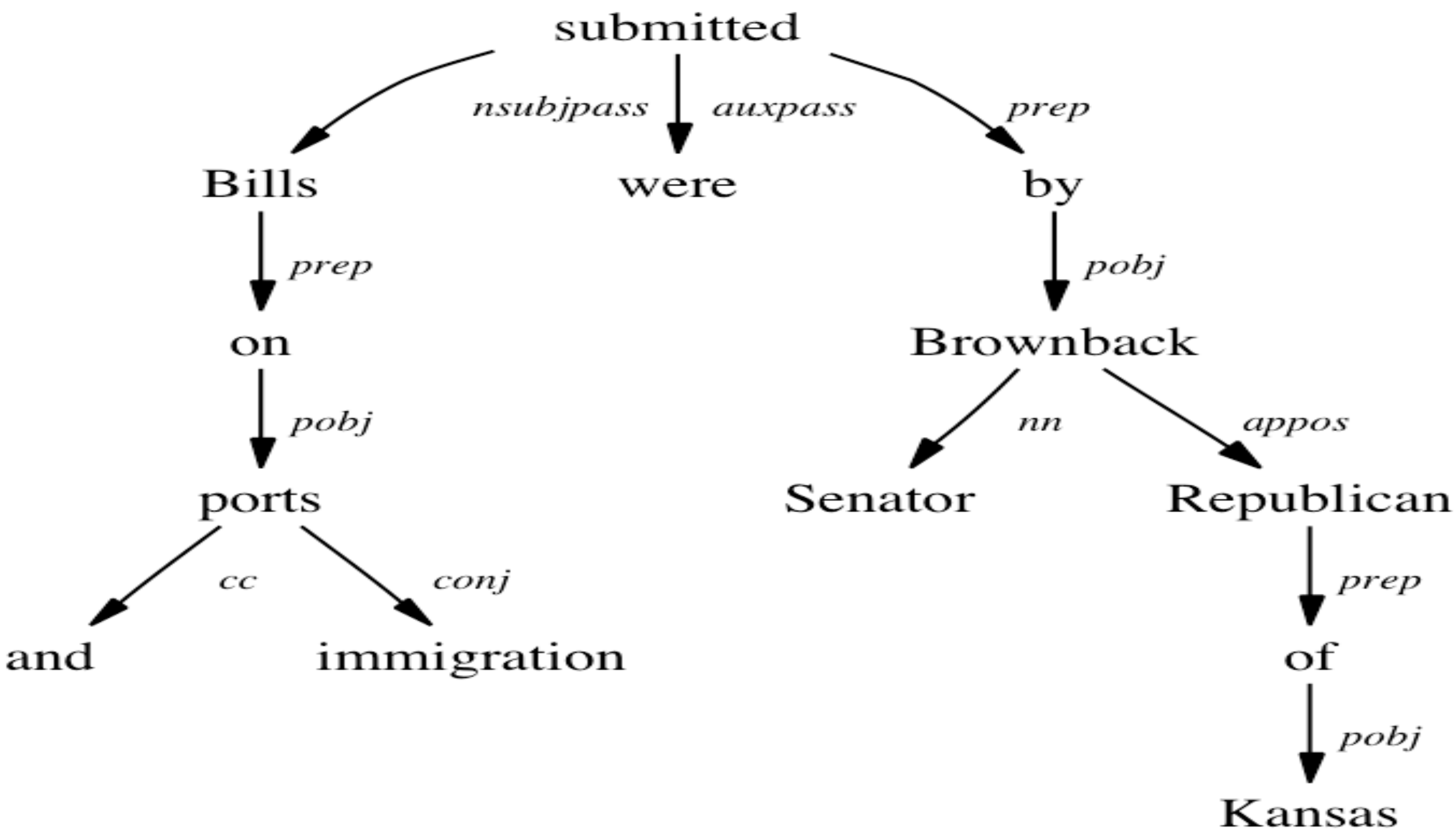
- Stanford Dependencies (SDs) provide a representation of grammatical relations between words in a sentence.
- Designed to be easily understood and effectively used by people who want to extract textual relations (and not only by linguists).
- SDs are triplets: name of the relation, governor and dependent.

Two Options

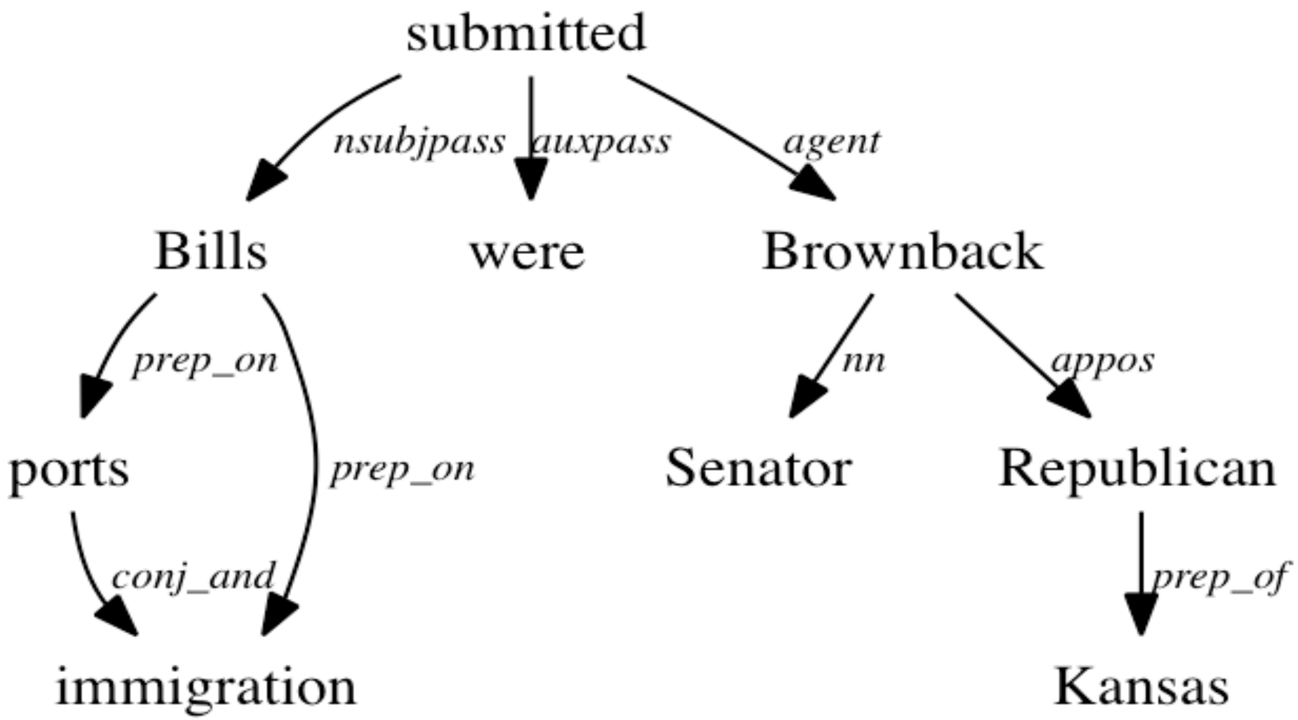
- every word of the original sentence is present as a node with relations between it and other nodes
  – close parallelism to the source text words
- certain words are “collapsed” out of the representation, e.g. turning prepositions into relations
  – more useful for relation extraction and shallow language understanding tasks.

Basic Dependency Representation

each word in the sentence (except the head of the sentence) is the dependent of one other word



Standard Dependencies
(collapsed and propagated)



Stanford Dependencies

- Initially produced using hand-written `tregex` patterns over English phrase-structure trees.
- Later available for Chinese and other languages, among them Italian.
- Now superseded by Universal Dependencies.

Universal Dependencies

- Cross-linguistically consistent grammatical annotation
- Support multilingual research in NLP and linguistics
  - Linguistic analysis within and across languages
  - Syntactic parsing in a monolingual and cross-lingual setting
  - Useful information for downstream applications
- Build on common usage and existing de facto standards

Material taken from Nivre's presentation at CLiC-it 2016
"Reflections on Universal Dependencies"

UD Treebanks

Ancient Greek	244K
Ancient Greek-PROIEL	206K
Arabic	242K
Basque	121K
Bulgarian	156K
Buryat	9K
Catalan	530K
Chinese	123K
Coptic	5K
Croatian	139K
Czech	1,503K
Czech-CAC	493K
Czech-CLTT	35K
Danish	100K
Dutch	209K
Dutch-LassySmall	98K
English	254K
English-ESL	97K
English-LinES	82K
Estonian	234K
Faroese	132K
Finnish	181K
Finnish-FTB	159K
French	391K
Galician	138K
Galician-TreeGal	24K
German	293K
Gothic	56K
Greek	59K
Hebrew	115K
Hindi	351K
Hungarian	42K
Indonesian	121K
Irish	23K
Italian	272K
Japanese	92K
Japanese-KTC	267K
Kazakh	6K
Latin	47K
Latin-ITTB	291K
Latin-PROIEL	165K
Latvian	20K
Norwegian-Bokmaal	310K
Old Church Slavonic	57K
Persian	151K
Polish	83K
Portuguese	209K
Portuguese-BR	298K
Portuguese-Bosque	227K
Romanian	218K
Russian	99K
Russian-SynTagRus	1,068K
Sanskrit	1K
Slovak	106K
Slovenian	140K
Slovenian-SST	29K
Spanish	423K
Spanish-AnCora	547K
Swedish	96K
Swedish-LinES	79K
Swedish Sign Language	<1K
Tamil	8K
Turkish	56K
Ukrainian	1K
Uyghur	6K
Vietnamese	43K

Stats of Dec 1, 2016:

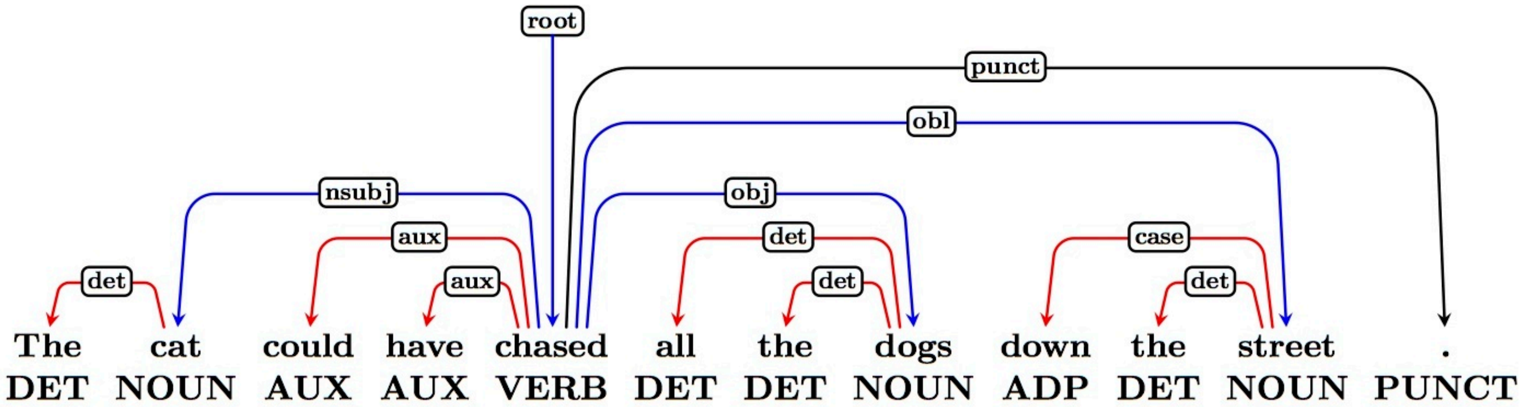
- 47 languages
- 64 treebanks
- 145 contributors
- 6803 downloads

Universal Dependencies

http://universaldependencies.org/

- Community effort (Stanford dependencies, Google universal POS tags, Interset interlingua for morphosyntactic tagsets)
- Universal taxonomy with language-specific elaboration
  - Languages select from a universal pool of categories
  - Allow language-specific extensions

Syntax



- Content words are related by dependency relations
- Function words attach to the content word they modify
- Punctuation attach to head of phrase or clause

Dependency Relations

- Taxonomy of 37 universal grammatical relations
  - Three types of structures: nominals, clauses, modifiers
  - Core arguments vs. other dependents (not complements vs. adjuncts)
  - Language-specific subtypes
- Basic and enhanced representations
  - Basic dependencies form a (possibly non-projective) tree
  - Additional dependencies in the enhanced representation

Dependency Relations

	Nominal	Clause	Modifier Word	Function Word
Core Predicate Dep	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Predicate Dep	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal Dep	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	parataxis list	orphan goeswith reparandum	punct root dep

* Generalized modifier of predicates and (non-nominal) modifiers

1. Context Free Grammars (CFGs)
2. Efficiency and Expressivity
3. Features and Unification
4. Dependency Grammars
5. **Resolving Ambiguity**
6. Treebanks and Evaluation

5. Resolving Ambiguity

- Ambiguity
- Probabilistic Context Free Grammars
  - Using PCFGs for disambiguation
- Training PCFGs
- Lexical preferences

Ambiguity

- lexical vs. structural

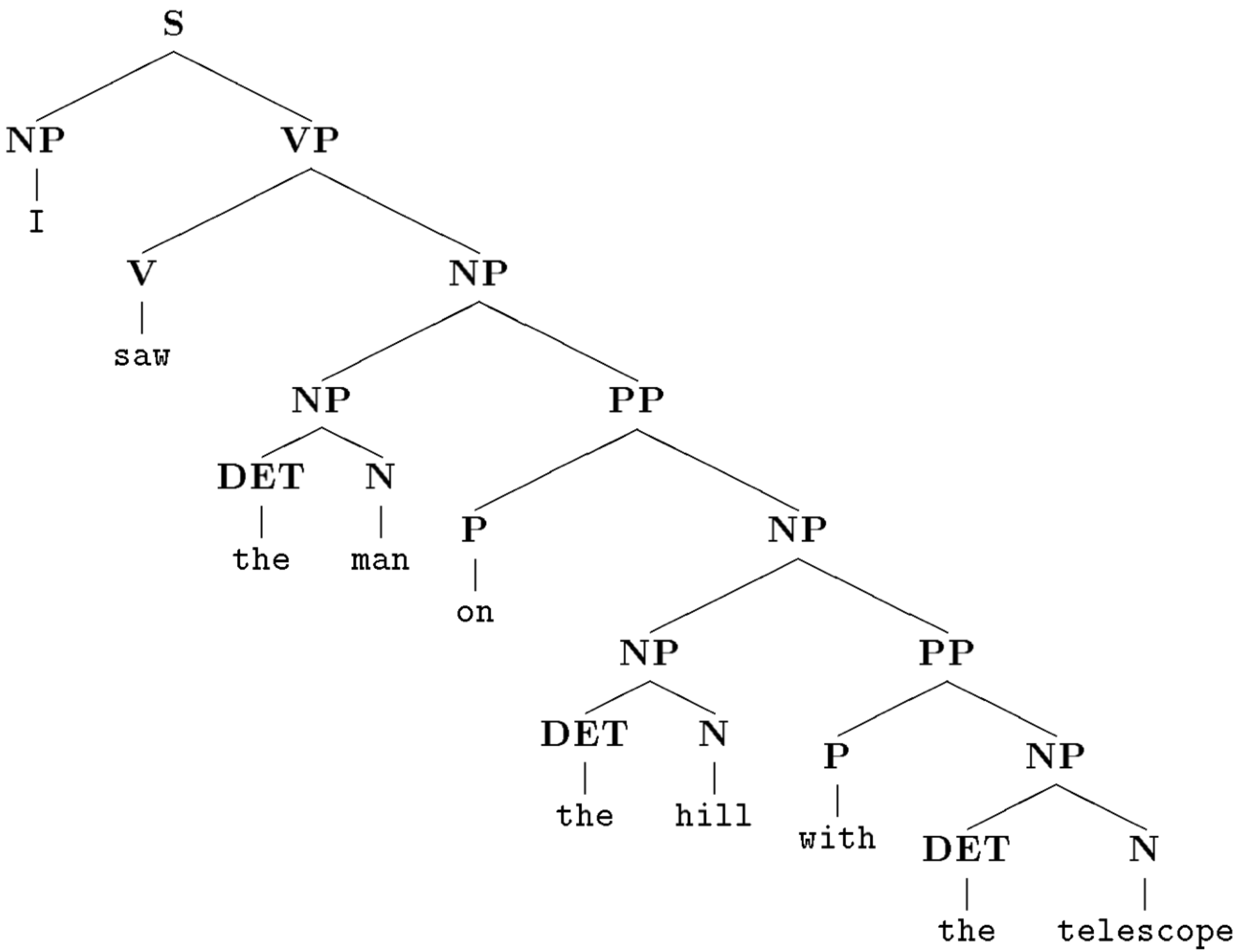
  lexical *fire* verb or noun?

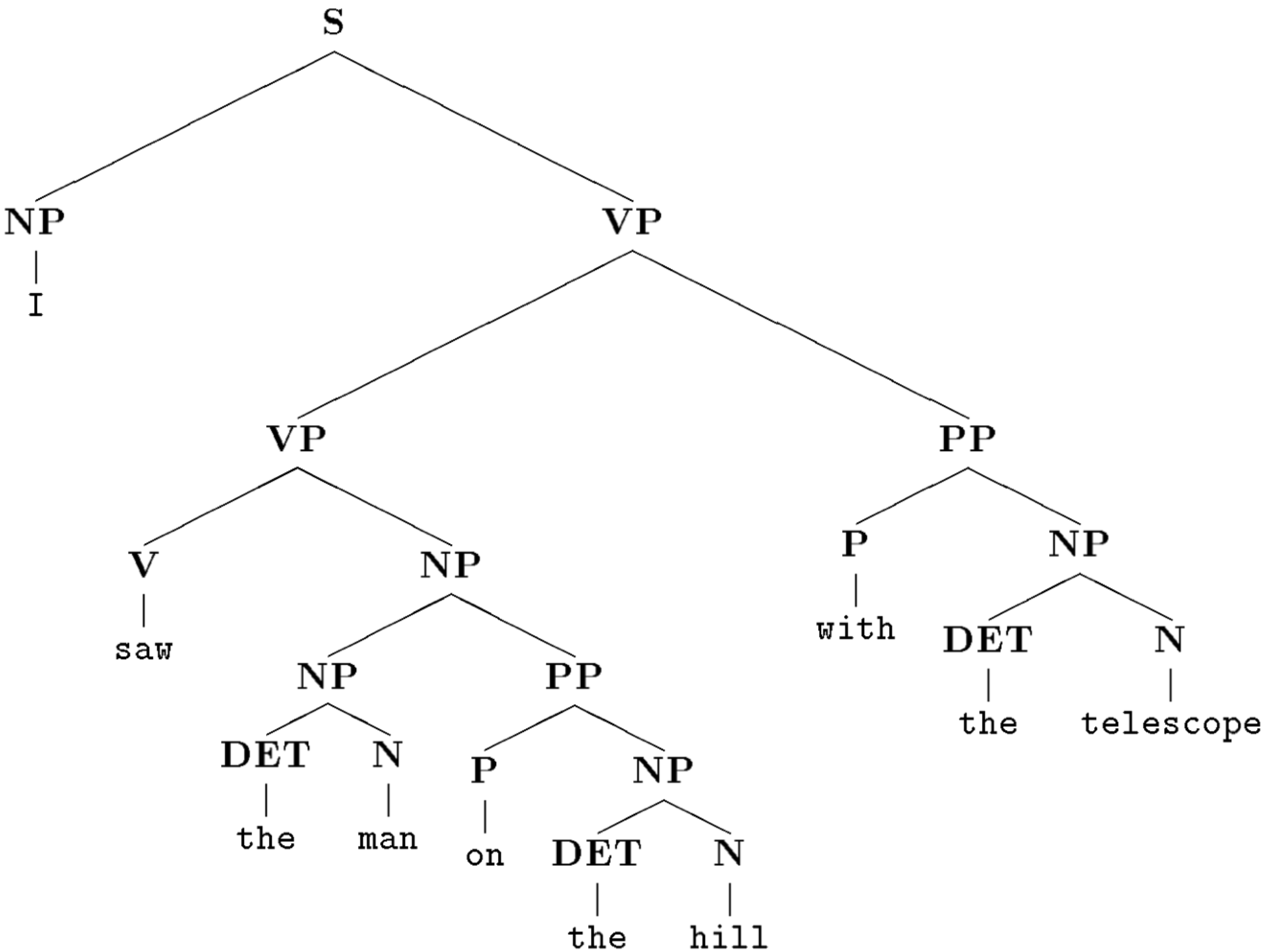
  if it is a verb, which sense?

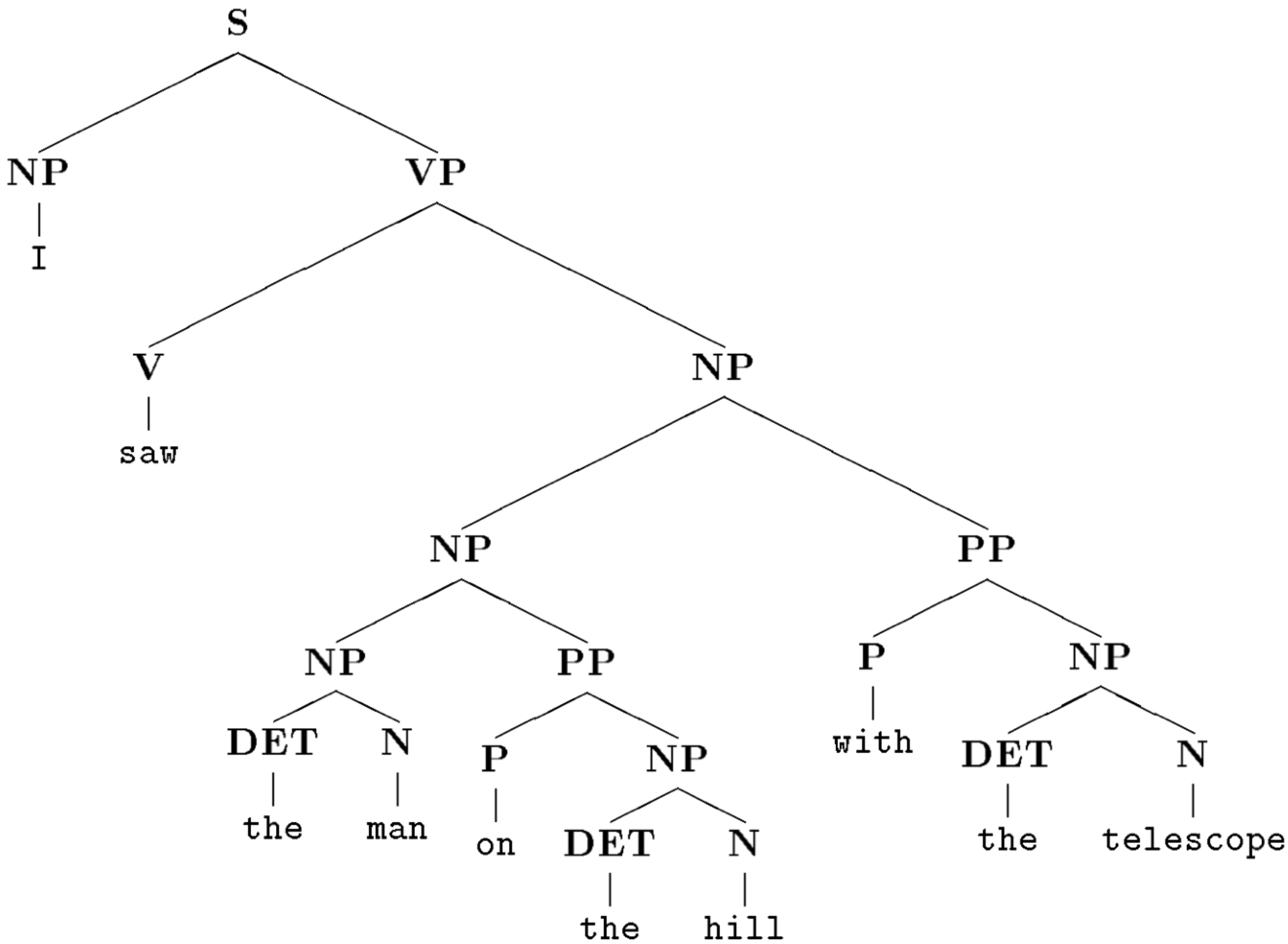
  `shoot`, `dismiss` or `burn`?

  structural

  I saw the man on the hill with the telescope


S
NP
VP
I
V
NP
saw
NP
PP
DET
N
P
NP
the
man
on
NP
PP
DET
N
P
NP
the
hill
with
DET
N
the
telescope


S
NP
I
VP
VP
V
saw
NP
NP
DET
the
N
man
PP
P
on
NP
DET
the
N
hill
PP
P
with
NP
DET
the
N
telescope


S
NP
I
VP
V
saw
NP
NP
NP
DET
the
N
man
PP
P
on
NP
DET
the
N
hill
PP
P
with
NP
DET
the
N
telescope

Ambiguity

- local vs. global

  local

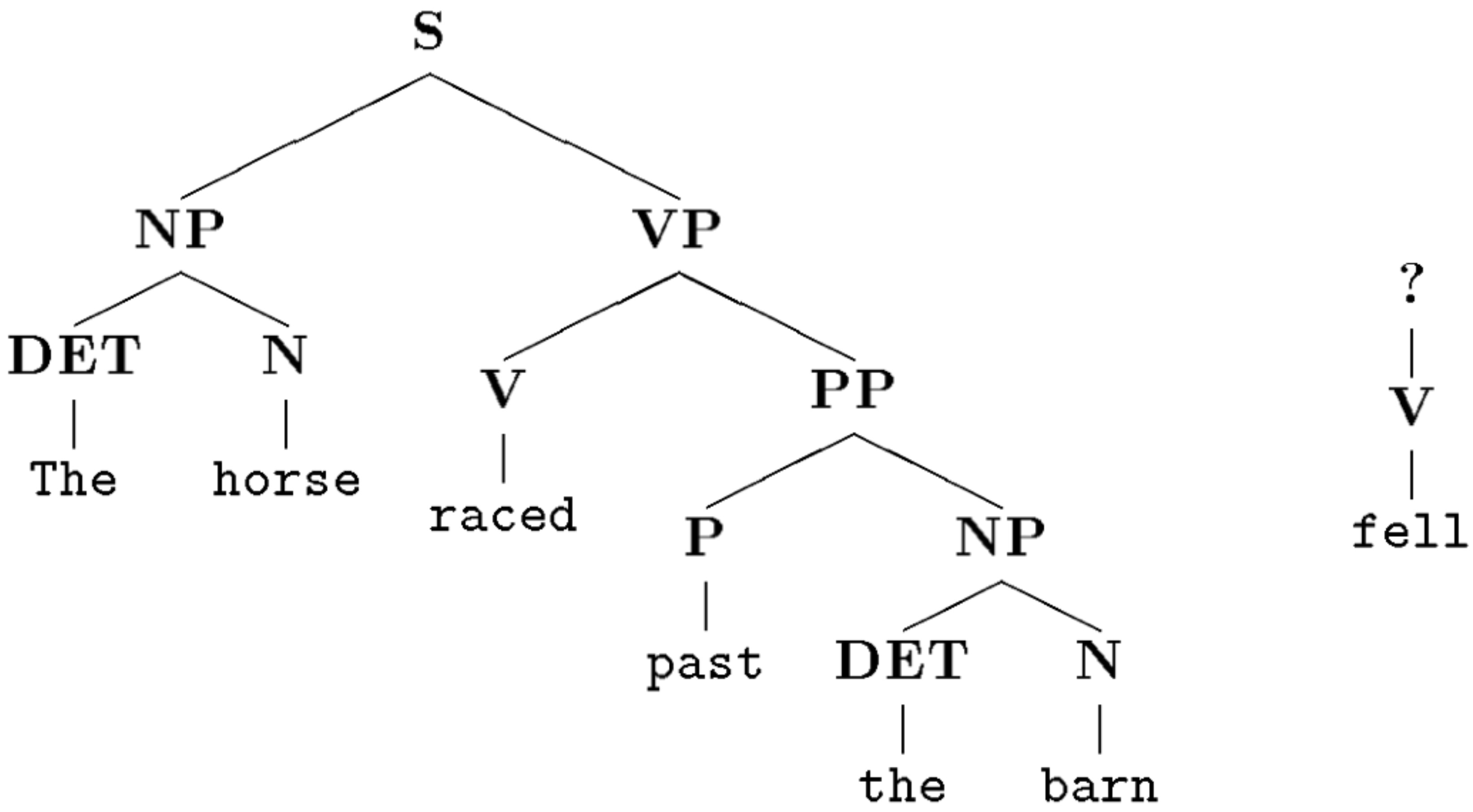
  garden path sentences:
  the horse raced past the barn fell

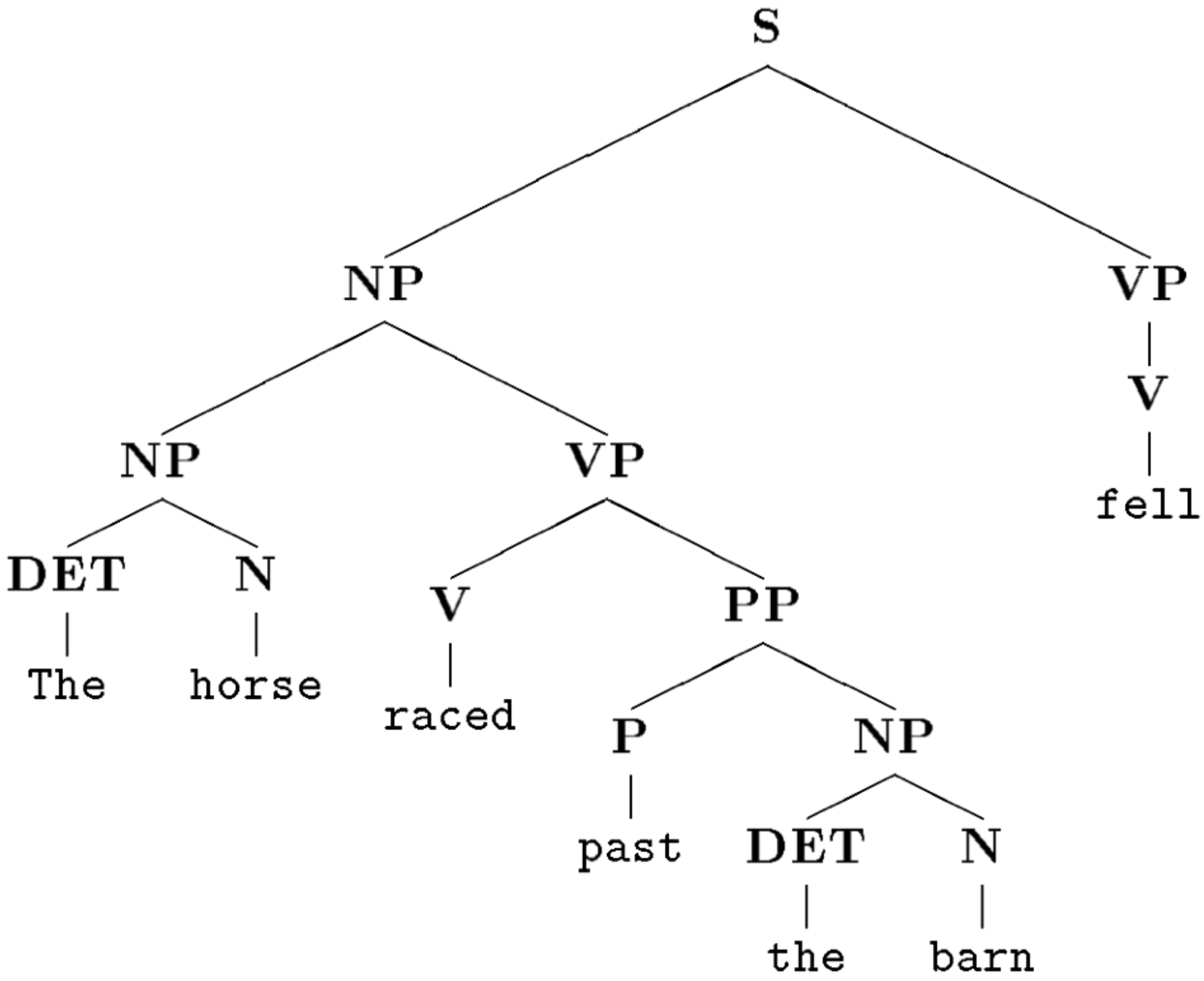
  – need backtracking, lookahead or parallelism

  global

  I saw the man on the hill with the telescope

  – need to solve the ambiguity using context


S
NP
VP
DET
N
V
PP
The
horse
raced
P
NP
past
DET
N
the
barn
?
V
fell


S
NP
VP
NP
VP
V
DET
N
V
PP
fell
The
horse
raced
P
NP
past
DET
N
the
barn

Ambiguity

- Not a phenomenon limited to “pathological” sentences but a pervasive feature of language
- Necessary to find effective ways to deal with it, particularly when we aim at providing robust parsers.

Probabilistic CFGs

A *PCFG* is a 5-tuple $G = (N, \Sigma, P, S, D)$, where D is a function assigning probabilities to each rule in P.

$$P(A \rightarrow \beta \mid A)$$

Considering all the possible expansions of a non-terminal, the sum of their probabilities must be 1.

Probability of a parse tree T on a sentence S:

$$P(T, S) = \Pi_{n \in T}\ p(r(n))$$

A Probabilistic CFG

$S \rightarrow NP\ VP$	1.0
$NP \rightarrow DT\ NN$	0.3
$NP \rightarrow NP\ PP$	0.7
NN → man	0.7
NN → woman	0.2
NN → telescope	0.1
DT → the	1.0

$VP \rightarrow Vi$	0.4
$VP \rightarrow Vt\ NP$	0.4
$VP \rightarrow VP\ PP$	0.2
Vi → laughs	1.0
Vt → saw	1.0
$PP \rightarrow P\ NP$	1.0
P → with	0.5
P → in	0.5

Probability of a tree with rules $\alpha_i \rightarrow \beta_i$:

$$\Pi_i P(\alpha_i \rightarrow \beta_i \mid \alpha_i)$$

Derivation	Rules used	Probability
S	S $\rightarrow$ NP VP	1.0
NP VP	NP $\rightarrow$ DT N	0.3
DT N VP	DT $\rightarrow$ the	1.0
the NP VP	N $\rightarrow$ man	0.7
the man VP	VP $\rightarrow$ Vi	0.4
the man Vi	Vi $\rightarrow$ laughs	1.0
the man laughs		

TOTAL PROBABILITY = $1.0 \times 0.3 \times 1.0 \times 0.7 \times 0.4 \times 1.0$

Properties of PCFGs

Given a sentence S, the set of derivations for that sentence is $\mathrm{T}(S)$. Then a PCFG assigns a probability to each element in $\mathrm{T}(S)$, so that parse trees can ranked in order of probability.

The probability of a sentence S is

$$\sum_{T \in \mathrm{T}(S)} \mathrm{P}(\mathrm{T}, \mathrm{S})$$

Learning Probabilistic CFGs

PCFGs can be learned from a treebank, i.e. a set of already parsed sentences.

Maximum Likelihood estimates of the probabilities can be obtained from the parse trees of the treebank:

$$P(A \rightarrow \beta \mid A) \approx \frac{Count(A \rightarrow \beta)}{\Sigma_{\gamma} Count(A \rightarrow \gamma)} = \frac{Count(A \rightarrow \beta)}{Count(A)}$$

Algorithms for PCFGs

Given a PCFG and a sentence *S*, T(*S*) be the set of trees with *S* as yield.

- Given a PCFG and a sentence *S*, how do we find

$$\arg\max_{T \in \mathcal{T}(S)} P(T, S)$$

- Given a PCFG and a sentence *S* , how do we find

$$P(S) = \sum_{T \in \mathcal{T}(S)} P(T, S)$$

Problems with PCFGs

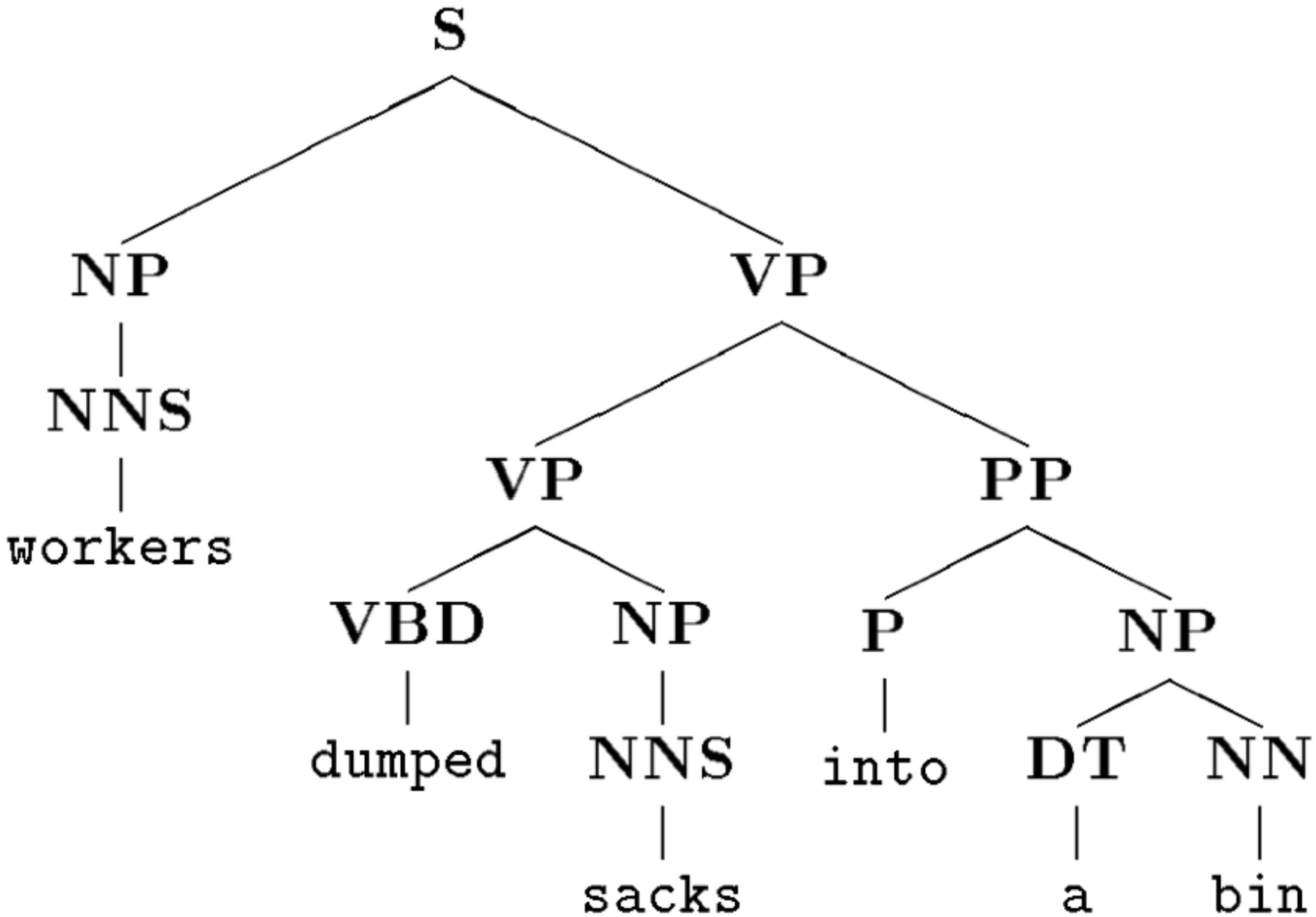
Problems in modeling
- *structural dependencies*
- *lexical dependencies*

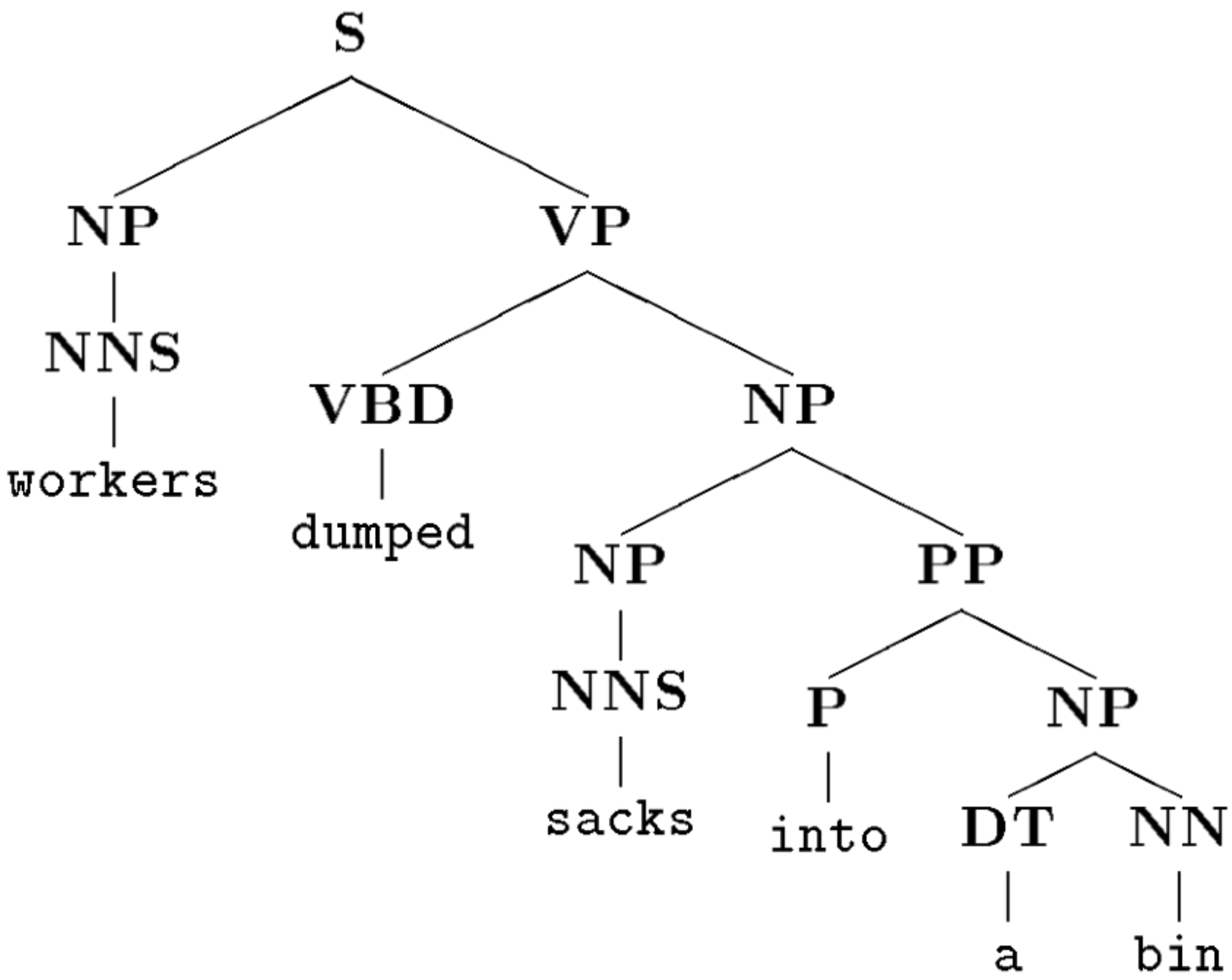
Independence assumption: expansion of any non-terminal is independent of the expansion of other non-terminal

Problems with PCFGs

- Lack of sensitivity to lexical choices (words).
- Importance of lexical information in selecting correct attachment of ambiguous PP-attachments.

PP-Attachment Ambiguity




S
NP
VP
NNS
workers
VBD
dumped
NP
NP
PP
NNS
sacks
P
into
NP
DT
NN
a
bin

The two parses differ only in one rule:

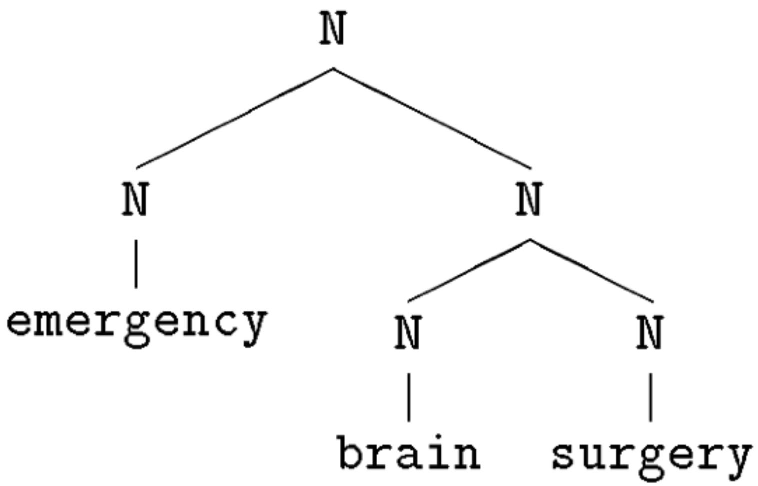
- VP $\rightarrow$ VP PP
- NP $\rightarrow$ NP PP

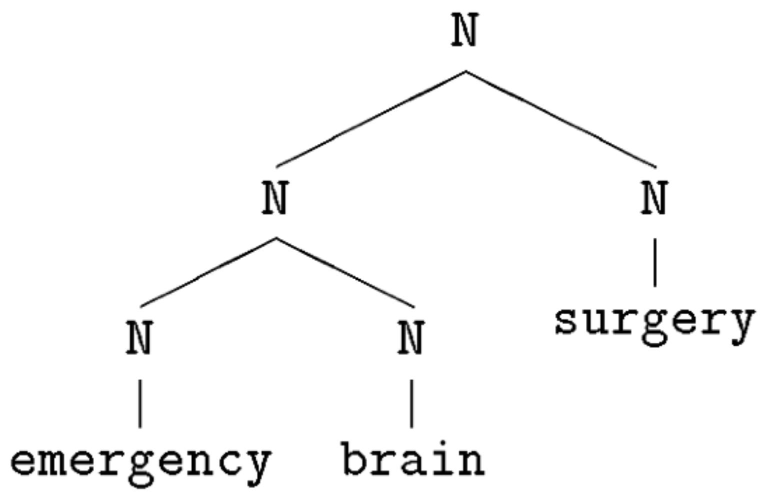
If $P(\text{VP} \rightarrow \text{VP PP} \mid \text{VP}) > P(\text{NP} \rightarrow \text{NP PP} \mid \text{NP})$ then the first parse is more probable; otherwise the second is more probable.

Attachment decision is
completely independent of the words

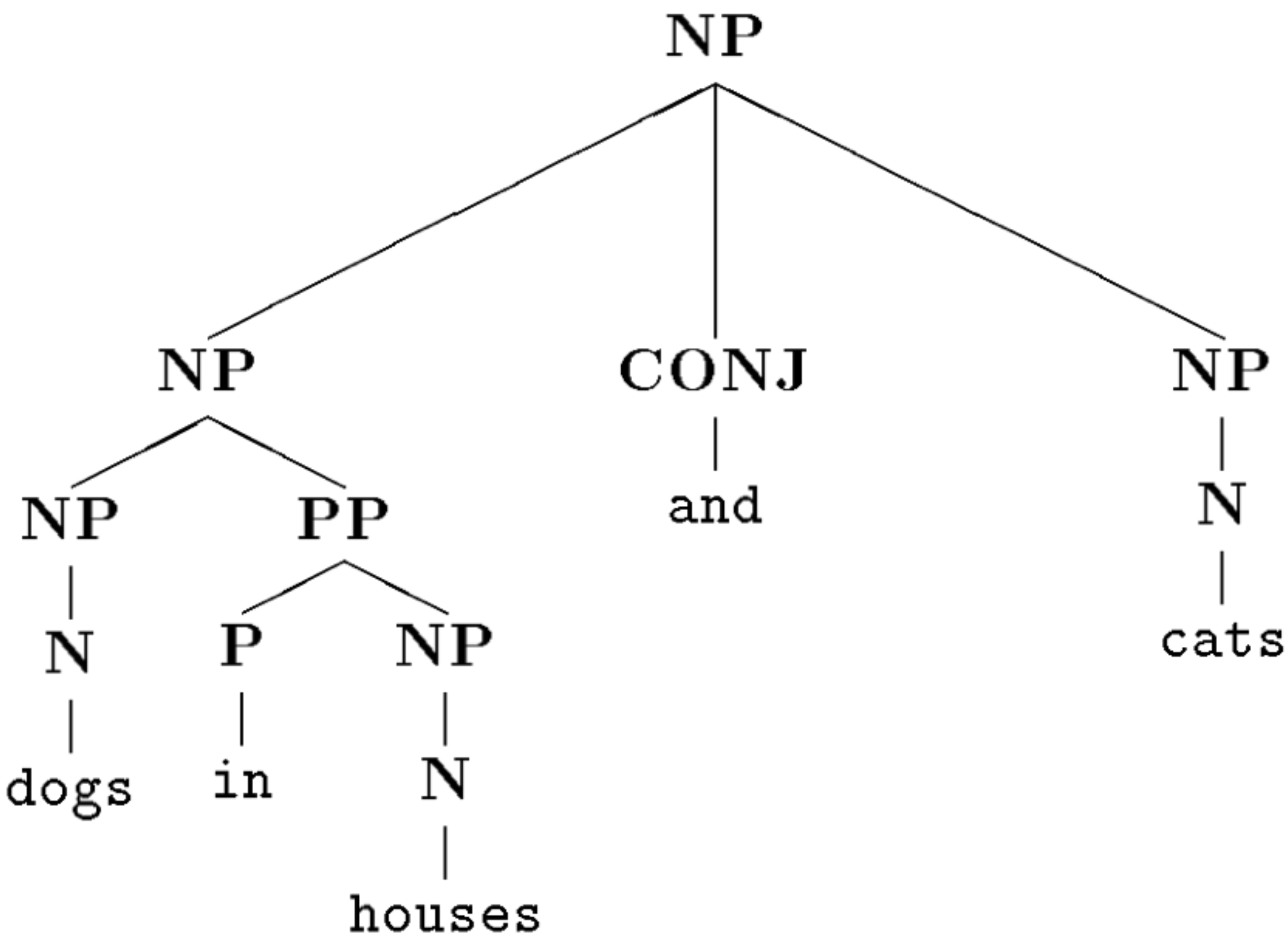
Problems with PCFGs

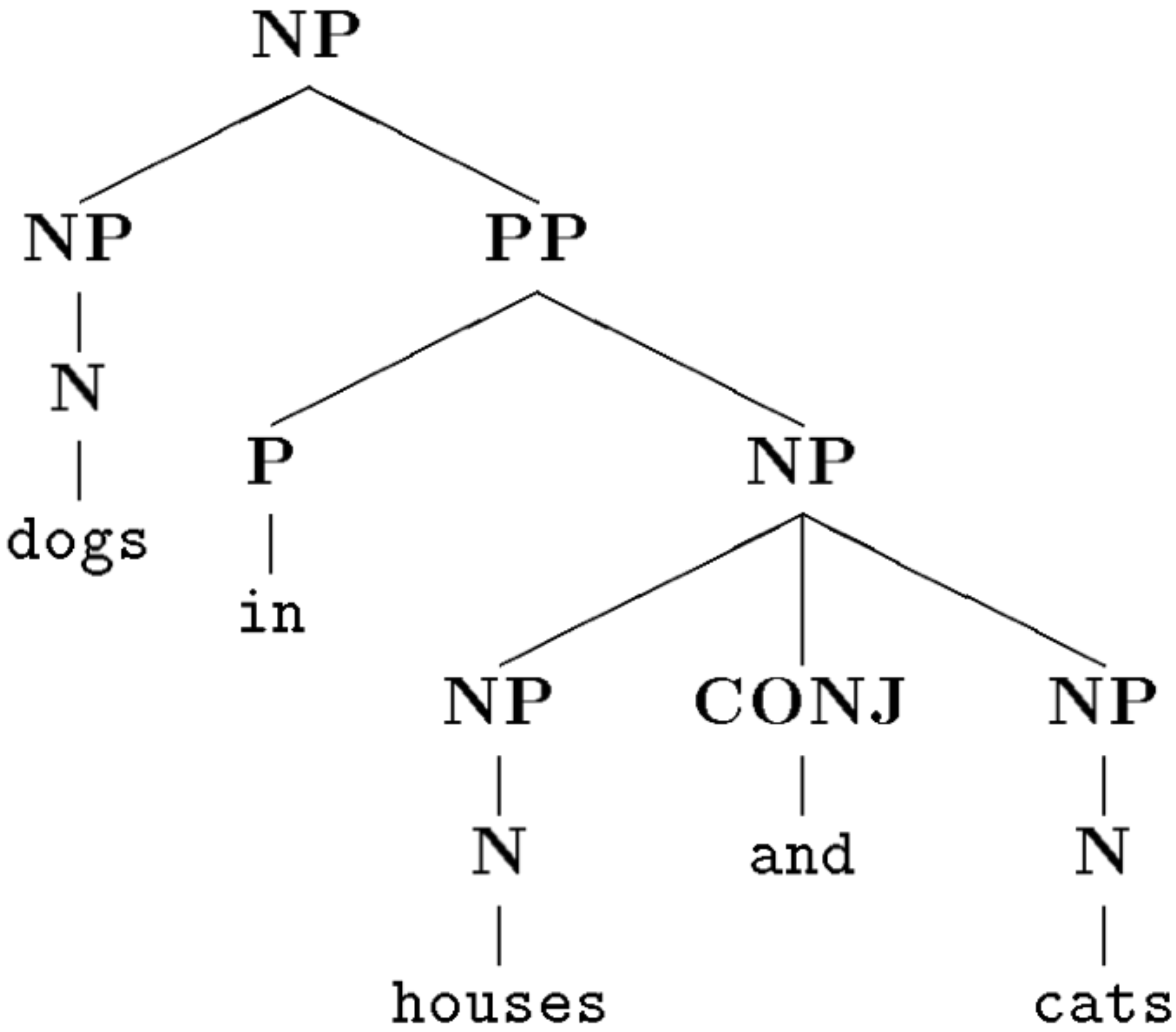
- A PCFG cannot distinguish between different derivations which use the same rules





Coordination Ambiguity

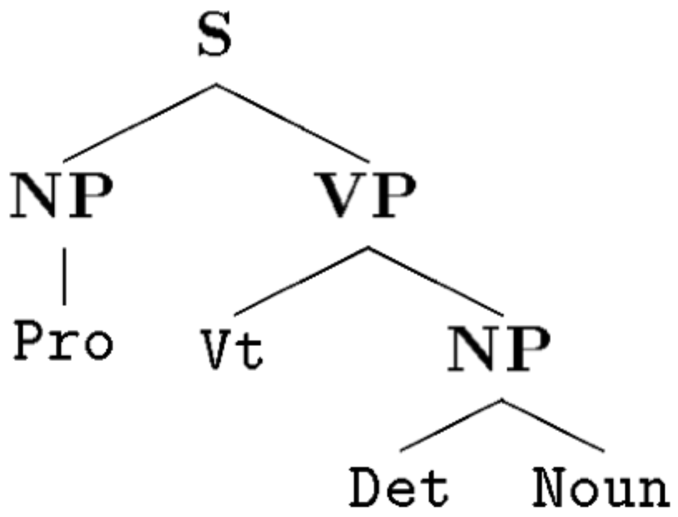


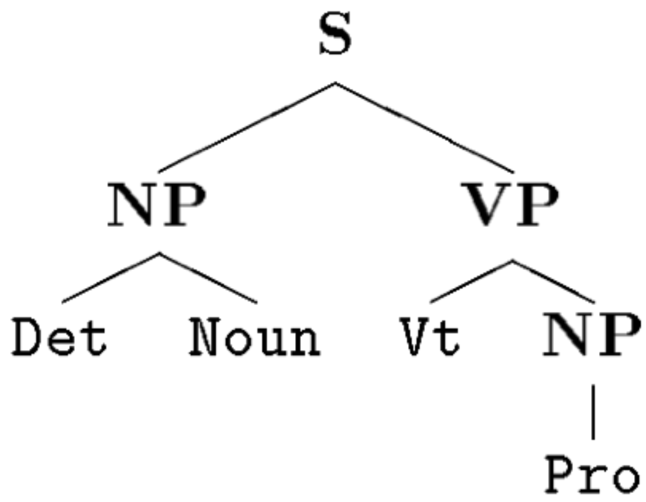

NP
NP
PP
N
dogs
P
in
NP
NP
CONJ
NP
N
and
N
houses
cats

The two parse trees have identical rules, and therefore have identical probabilities under any assignment of PCFG rule probabilities.

Problems with PCFGs

- Probabilities of sub-trees cannot be dependent on context. E.g., a pronoun is relatively more common as a subject than as an object in a sentence, but a single rule $NP \rightarrow Pro$ cannot account for this fact.


S
NP
VP
Pro
Vt
NP
Det
Noun


S
NP
VP
Det
Noun
Vt
NP
Pro

Lexicalized PCFGs

A lexical *head* is associated to each syntactic constituent.

Each PCFG rule is augmented to identify one right-hand side constituent as its *head* daughter.

$$p(r(n) \mid n, h(n))$$

Problems with data sparseness: need to smooth to avoid 0 probabilities.

Data Sparseness

Use of lexical information enlarges an already existing problem: in WSJ, 15% of all test data sentences contain a rule never seen in training.

We'll see later how to deal with data sparseness.

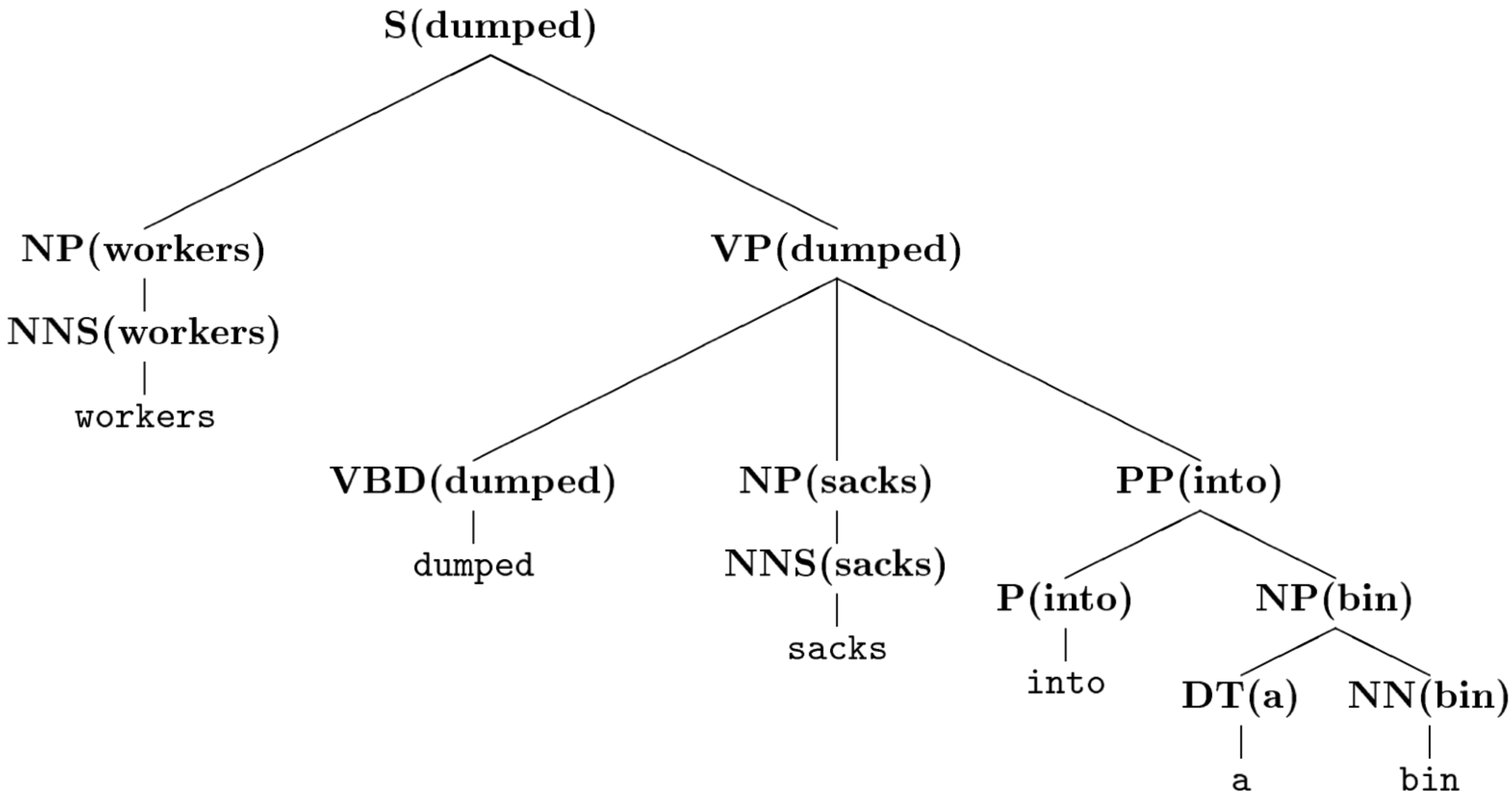
Lexicalized PCFGs

- Each PCFG rule is augmented to identify one right-hand side constituent as its head daughter.
  - S → NP VP (VP is the head)
  - VP → Vt NP (Vt is the head)
  - NP → DT NN (NN is the head)
- A core idea in linguistics (Dependency Grammar, X-bar Theory, Head-Driven Phrase Structure Grammar)

Rules for identifying heads

- Need a way to identify heads in the rules
- There are good linguistics criteria… unfortunately they don't always work with real-world grammars extracted from treebanks
- Need of integrating linguistic criteria with hacks which rely on the idiosyncrasies of the treebank

Adding Headwords to Trees



Adding Headwords to Rules

We can estimate probabilities for lexicalized PCFGs as for simple PCFGs.

- VP(dumped) → VBD(dumped) NP(sacks) PP(into)
- …

However, this produces an increase in the number of rules and a problem of data sparseness because no treebank is big enough to train such probabilities.

Adding Headwords to Rules

Need some simplifying independence assumptions in order to cluster some of the counts

Statistical parsers (e.g., Charniak, Collins) usually differ in the independence assumptions they make

Headwords and Dependencies

A new representation: a tree is represented as a set of *dependencies*, not as a set of *context-free rules*

A **dependency** is a 8-tuple:

- headword
- headtag
- modifier word
- modifier tag
- parent non-terminal
- head non-terminal
- modifier non-terminal
- direction

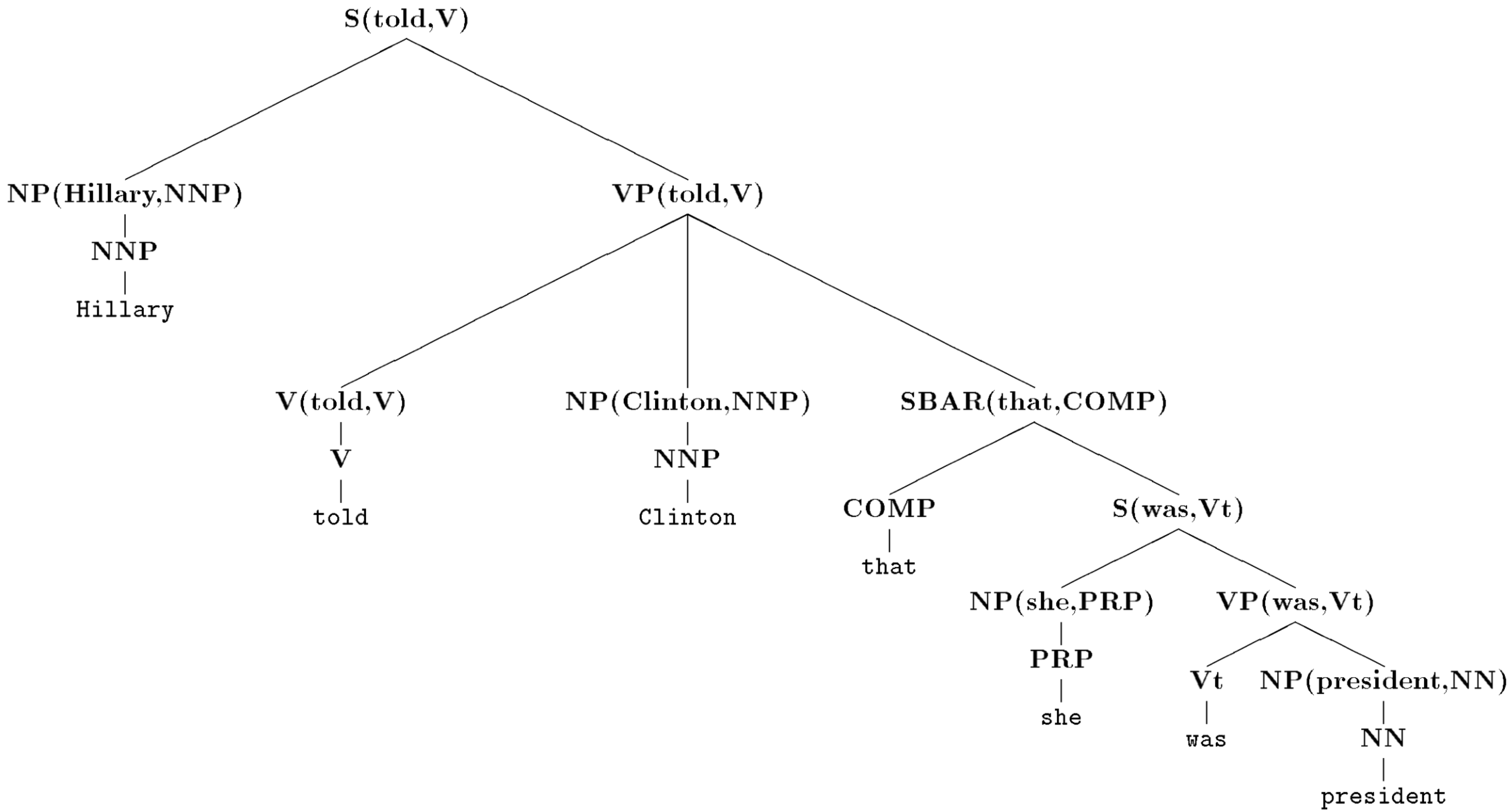
Headwords and Dependencies

Each rule with n children contributes
(n - 1) dependencies:

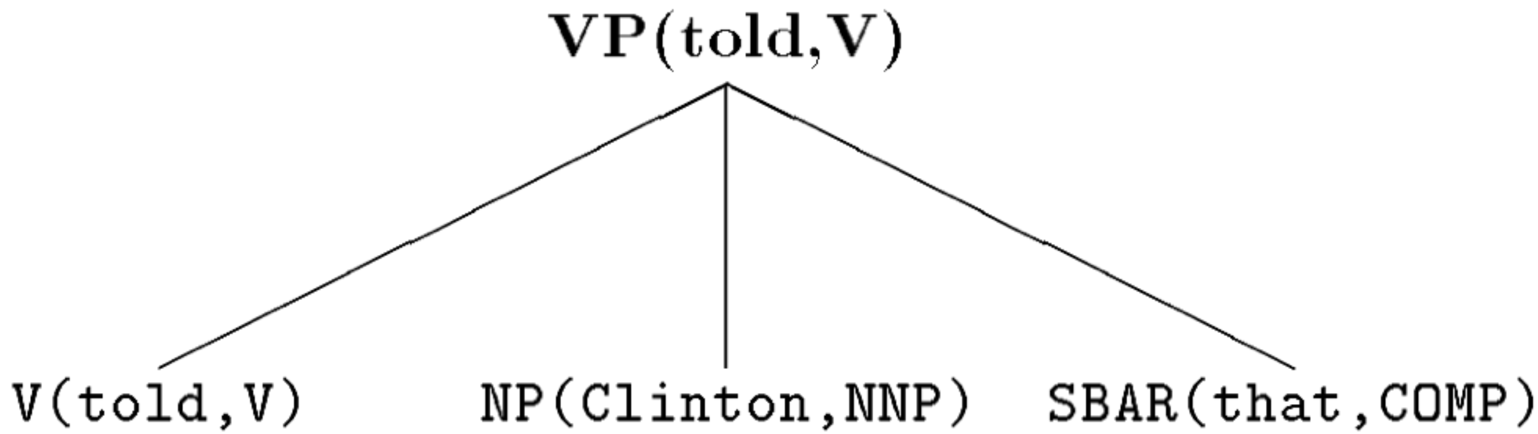
$$\textbf{VP(dumped,VBD)} \Rightarrow \textbf{VBD(dumped,VBD) NP(sacks,NNS)}$$

$$\Downarrow$$

(dumped, VBD, sacks, NNS, VP, VBD, NP, RIGHT)

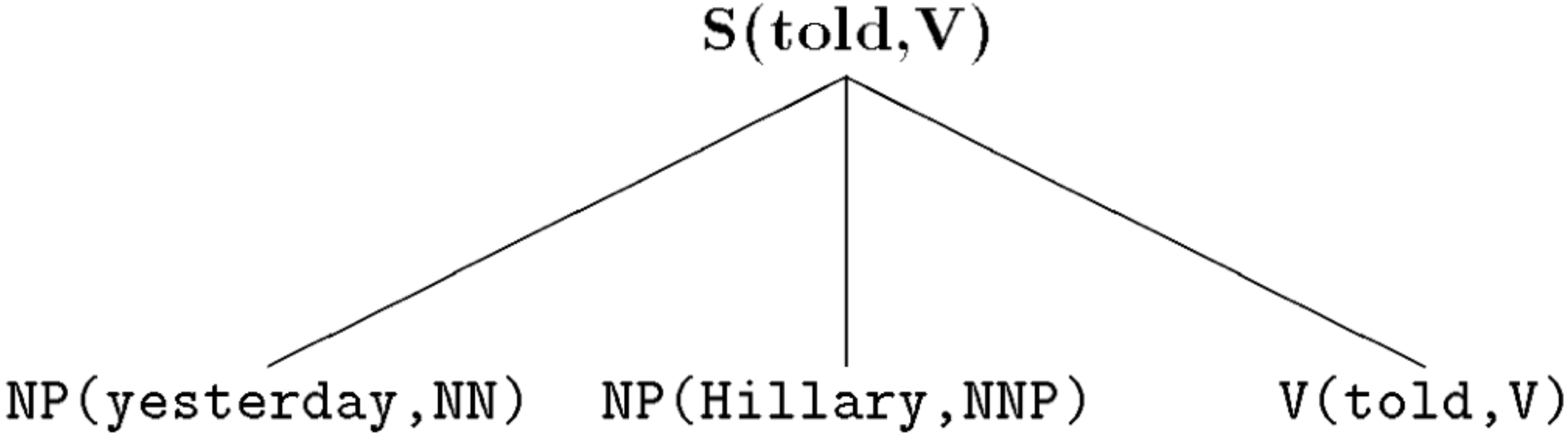

S(told,V)
NP(Hillary,NNP)
NNP
Hillary
VP(told,V)
V(told,V)
V
told
NP(Clinton,NNP)
NNP
Clinton
SBAR(that,COMP)
COMP
that
S(was,Vt)
NP(she,PRP)
PRP
she
VP(was,Vt)
Vt
was
NP(president,NN)
NN
president

Headwords and Dependencies



(told, V, Clinton, NNP, VP, V, NP, RIGHT)
(told, V, that, COMP, VP, V, SBAR, RIGHT)

Headwords and Dependencies



(told, V, yesterday, NN, S, VP, NP, LEFT)
(told, V, Hillary, NNP, S, VP, NP, LEFT)

Smoothed Estimation

We need to perform some kind of smoothing to avoid 0 probabilities. E.g.:

$$e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2)\, e_3)$$

where e_1, e_2 and e_3 are maximum likelihood estimates with different contexts and λ_1, λ_2 and λ_3 are smoothing parameters where $0 \leq \lambda_i \leq 1$

Constituency Parsers (1)

- Michael Collins
  http://www.cs.columbia.edu/~mcollins/code.html
  available as Solaris/Linux executable w/o the possibility of retraining on different corpora
- Dan Bikel's Multilingual Statistical Parsing Engine
  http://www.cis.upenn.edu/~dbikel/software.html#stat-parser
  Java reimplementation of Collins' parser, highly customizable to new corpora and new languages (English, Chinese, Arabic, Italian)

Constituency Parsers (2)

- Stanford parser
  http://nlp.stanford.edu/software/lex-parser.shtml
  a Java implementation of probabilistic natural language parsers, both highly optimized PCFG and dependency parsers, and a lexicalized PCFG parser (applied to English, Chinese, German, Italian, Arabic)

Constituency Parsers (3)

- Berkeley parser
  https://github.com/slavpetrov/berkeleyparser
  - state-of-the-art for English on the Penn Treebank (*)
  - outperforms other parsers in languages different from English (e.g. German, Chinese, French)
  - no need of language-specific adaptations
  - written in Java
  - based on a hierarchical coarse-to-fine parsing, where a sequence of grammars is considered, each being the refinement, i.e. a partial splitting, of the preceding one.

Constituency Parsers (4)

- Charniak-Johnson reranking parser
  http://bllip.cs.brown.edu/resources.shtml
  - state-of-the-art for English on the Penn Treebank
  - based on two steps: the former generates the *N* best analyses, the latter reranks them using various features.

Dependency Parsers (1)

- MaltParser
  http://maltparser.org/
  a language-independent system for data-driven dependency parsing written in Java (open source).
  Applied to Bulgarian, Chinese, Czech, Danish, Dutch, English, German, Italian, Swedish, Turkish.
- DeSR Dependency Parser
  http://desr.sourceforge.net
  a shift-reduce dependency parser (open source).
  Applied to Bulgarian, Chinese, Czech, Danish, Dutch, English, German, Italian, Swedish, Turkish.
  Online demo: http://tanl.di.unipi.it/en/

Dependency Parsers (2)

- MATE Parsers
  http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/matetools.en.html
- TurboParser
  http://www.cs.cmu.edu/~ark/TurboParser/
- NLP4J Parser
  https://emorynlp.github.io/nlp4j/components/dependency-parsing.html
- RBG Parser
  https://github.com/taolei87/RBGParser

Dependency Parsers (3)

Complete pipelines:

- UDPipe
  https://ufal.mff.cuni.cz/udpipe
- Google SyntaxNet
  https://github.com/tensorflow/models/tree/master/syntaxnet

Online Demos

- http://nlp.mathcs.emory.edu:8080/nlp4j/NLP4JServlet
- http://nlp.stanford.edu:8080/parser/
- http://demo.ark.cs.cmu.edu/parse

1. Context Free Grammars (CFGs)
2. Efficiency and Expressivity
3. Features and Unification
4. Dependency Grammars
5. Resolving Ambiguity
6. **Treebanks and Evaluation**

6. Treebanks and Evaluation

- Treebanks (PennTreeBank)
- Evaluation of parsers
- EVALITA parsing tasks

Treebanks

- Set of sentences where each sentence is associated with the corresponding linguistic information:
  - Part of speech (PoS) tags (e.g., N=noun, V=verb)
  - parse trees
- Initially available for English only.
- Currently there are treebanks for several different languages (e.g., Chinese, Czech, German, French, Arabic, Italian, …).

Penn Treebank

- Penn Treebank (PTB): a large corpus of American English texts annotated both with PoS tags and with parse trees.
- Wall Street Journal (WSJ): the PTB subset usually adopted to test parsers' performance

Wall Street Journal

- Wall Street Journal (WSJ):
  - more than 1 million words
  - automatically annotated and manually corrected
  - divided into sections (00-24)
  - training: sections 02-21
  - validation: section 00
  - test: section 23
  - sentences of length < 40

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```

Parser Evaluation

- evaluation of bracketing accuracy in a test-file against a gold-file
- PARSEVAL performance measures (Black et al. 1991): labeled precision, labeled recall, crossing brackets
- a constituent in a candidate parse *c* of sentence *s* is labeled "correctly" if there is a constituent in the treebank parse with same starting point, ending point and non-terminal symbol.

Parser Evaluation

$$\text{labeled recall} := \frac{\#\,\text{correct constituents in candidate parse of } s}{\#\,\text{correct constituents in treebank parse of } s}$$

$$\text{labeled precision} := \frac{\#\,\text{correct constituents in candidate parse of } s}{\#\,\text{total constituents in candidate parse of } s}$$

cross-brackets: the number of crossed brackets (e.g. the number of constituents for which the treebank has a bracketing such as ((A B) C) but the candidate parse has a bracketing such as (A (B C)))

Parser Evaluation

- best results on WSJ:
  - about 90% for both precision and recall and about 1% cross-bracketed constituents
  - reranking parsers: about 92%

CoNLL Shared Tasks

Dependency parsing:

- CoNLL 2006 – Multilingual parsing: Arabic, Bulgarian, Chinese, Czech, Danish, Dutch, German, English, Japanese, Polish, Slovene, Spanish, Swedish, Turkish
- CoNLL 2007:
  - Multilingual track: Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, Turkish
  - Domain Adaptation track

EVALITA (http://www.evalita.it)

- Evaluation of NLP Tools for Italian
- First edition
  - Evaluation: May 2007
  - Workshop: September 2007
- Second edition
  - Evaluation: September 2009
  - Workshop: December 2009
- Third edition
  - Evaluation: October 2011
  - Workshop: January 2012

EVALITA 2007

- Five tasks:
  - PoS Tagging
  - Parsing
  - WSD
  - Temporal Expressions
  - Named Entities

EVALITA 2009

- Five text tasks:
  - PoS Tagging
  - Parsing
  - Lexical Substitution
  - Entity Recognition
  - Textual Entailment
- Three speech tasks:
  - Connected Digits Recognition
  - Spoken Dialogue Systems Evaluation
  - Speaker Identity Verification

EVALITA 2011

- Text tasks:
  - Parsing
  - Named Entity Recognition on Transcribed Broadcast News
  - Cross-Document Coreference Resolution
  - Anaphora Resolution
  - Super Sense Tagging
  - Frame Labeling over Italian Texts
  - Lemmatisation
- Speech tasks:
  - Automatic Speech Recognition - Large Vocabulary Transcription
  - Forced Alignment on Spontaneous Speech
  - Voice Applications on Mobile - Student Contest

EVALITA 2014 Parsing Task

- http://www.evalita.it/2014/tasks/dep_par4IE
- Dependency Parsing task
  based on the newly developed “Italian Stanford Dependency Treebank” (ISDT)

EVALITA 2014 Parsing Task

Three main novelties:

- the size of the dataset, much bigger than the resources used in the previous EVALITA campaigns;
- the annotation scheme, compliant to *de facto* standards at the level of both representation format (CoNLL) and adopted tagset (Stanford Dependencies);
- oriented to supporting IE tasks, a feature inherited from Stanford Dependencies.

Italian SD Treebank

- Italian resource annotated according to Stanford Dependencies.
- Obtained through a semi-automatic conversion process starting from MIDT.
- MIDT in turn was obtained merging two existing Italian treebanks: TUT and ISST-TANL.

EVALITA 2014 Parsing Task

Two subtasks:

- basic task on standard dependency parsing of Italian texts: double evaluation track aimed at testing the performance of parsing systems as well as their suitability to IE tasks;
- pilot task on cross-lingual transfer parsing: a parser trained on ISDT (universal version) is used on test sets of other (not necessarily typologically related) languages.

Open Issues - 1

How well does a parser trained on a given corpus work when applied to a different corpus?
E.g., training on WSJ and testing on a different treebank (e.g., the Brown corpus) results in a considerable drop in performance.

Open Issues - 2

How much do the proposed approaches rely on specificities of a given treebank and/or of a given language (usually English)?

- Training and testing on the Brown corpus produces worse results than on WSJ.
- On languages other than English performance is worse (sometimes considerably worse).
- Language-independent approaches are emerging.

Open Issues - 3

What happens if we want to work on languages without (or with a limited amount of) resources?

Performance crucially relies on the availability of sufficiently large treebanks.