GRAMMARS & PARSING
Probabilistic parsing & corpora
Dependency grammars & parsing

Natural Language Processing

Ing. R. Tedesco. PhD, AA 20-21

Probabilistic CFG (PCFG)

- The probabilistic model
  - Assigns probabilities to parse trees
- Takes into account probabilities in the model
- Parsing with probabilities
  - Simple adaptations of the dynamic programming algorithms
  - The goal is to find the tree associated with the maximum probability

The probabilistic model

- Adds probabilities to grammar rules
- Summing probabilities associated to the set of rules expanding the same non-terminal symbol, the result is 1

$$VP \rightarrow Verb \quad .55 = P(Verb \mid VP)$$

$$VP \rightarrow Verb\ NP \quad .40 = P(Verb\ NP \mid VP)$$

$$VP \rightarrow Verb\ NP\ NP \quad .05 = P(Verb\ NP\ NP \mid VP)$$

$$\overline{}$$

1

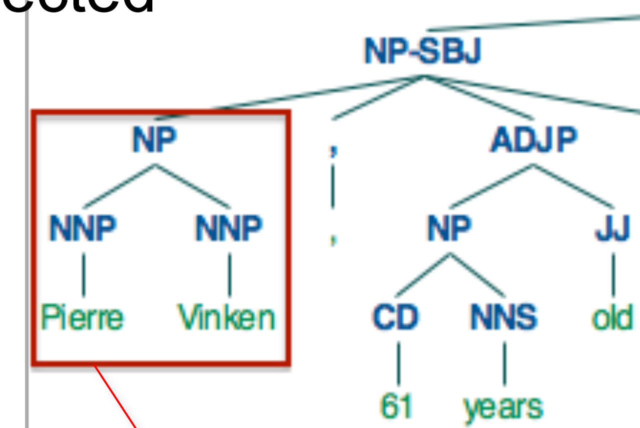
The probabilistic model

- The parse tree defines the grammar rules
- Parse tree probability
  - Calculated as the product of probabilities associated to the rules involved in tree derivation
- Probability of a word sequence (phrase)
  - Is the probability of the associated parse tree, if no ambiguities exist
  - Is the sum of probabilities associated to parse trees, if ambiguities exist
- Probabilities are calculated by means of an annotated database (treebank):

$$P(a \rightarrow b) = P(b \mid a) = \frac{C(a \rightarrow b)}{\sum_{\gamma} C(a \rightarrow \gamma)} = \frac{C(a \rightarrow b)}{C(a)}$$

Parsed Corpora: Treebanks

- The Penn Treebank
- Treebanks can be used to derive a CFG/PCFG
  - CFG: just collect the set of productions used in the treebank
  - PCFG: add probabilities to the collected productions, as explained before
- Issues:
  - The grammar generalize poorly to sentences that are not included in the treebank



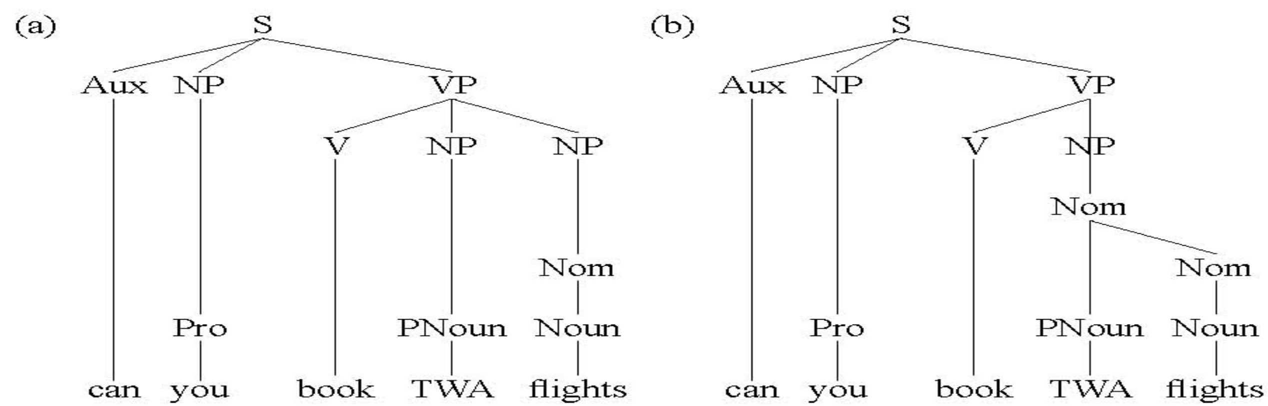
NP → NNP NNP
NNP → Pierre
NNP → Vinken

Probabilistic Parsing

- Probabilistic parsing needs
  - A grammar
  - A vast and robust dictionary with POS
  - A parser
- Dynamic programming algorithm: CKY (Cocke-Kasami-Younger) – Ney91 – Collins99 – Aho & Ullman72
  - Assigns probabilities to constituents when they are completed and put in the table
  - Bottom-up parser: uses maximum probability for going towards the top
- Other dynamic programming algorithm: Viterbi parsing

Example

- $P(T_{(a)}) = 1.5\times10^{-6}$
- $P(T_{(b)}) = 1.7\times10^{-6}$



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

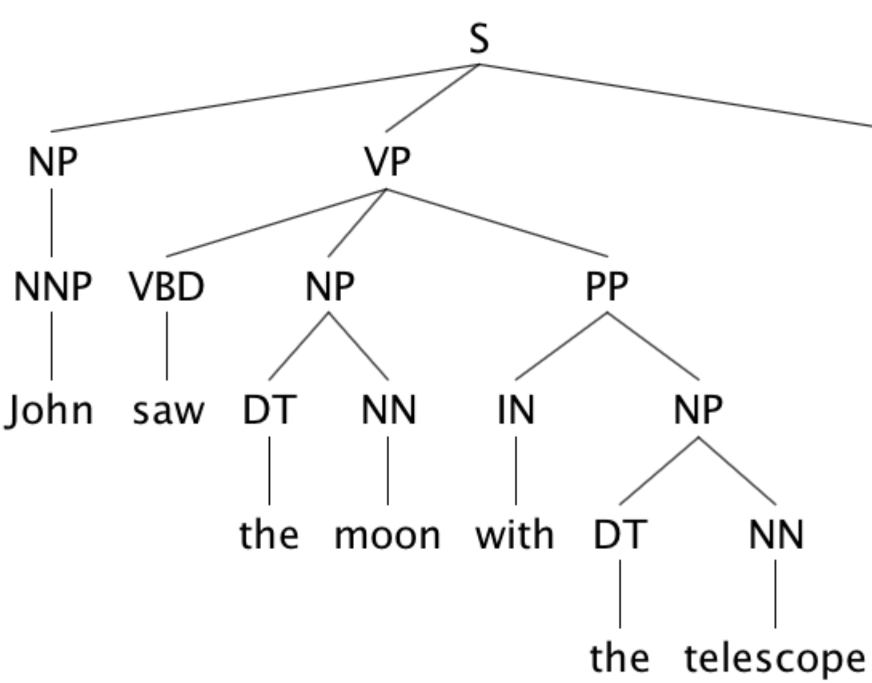
PCFGs: problems

- CFG assumption
  - Expansions of non-terminal symbols are independent. PCFG retains such assumption (i.e., probabilities can be multiplied ...)
- Problems related to:
  - **Structural dependencies** – Francis et al. 99
    - For **subjects NP**:
      NP→ Pronoun (91%), *NP→ Det Noun* (9%)
    - For **direct objects NP**:
      NP→ Pronoun (34%), *NP→ Det Noun* (66%)
    - But PCFG is not aware of this!
  - **Lexical dependencies** – example: PP attachment
    - *John saw the **moon** with the **telescope***
    - *John saw the **man** with the **hat***
    - Same structural form, only words change… see next slide...

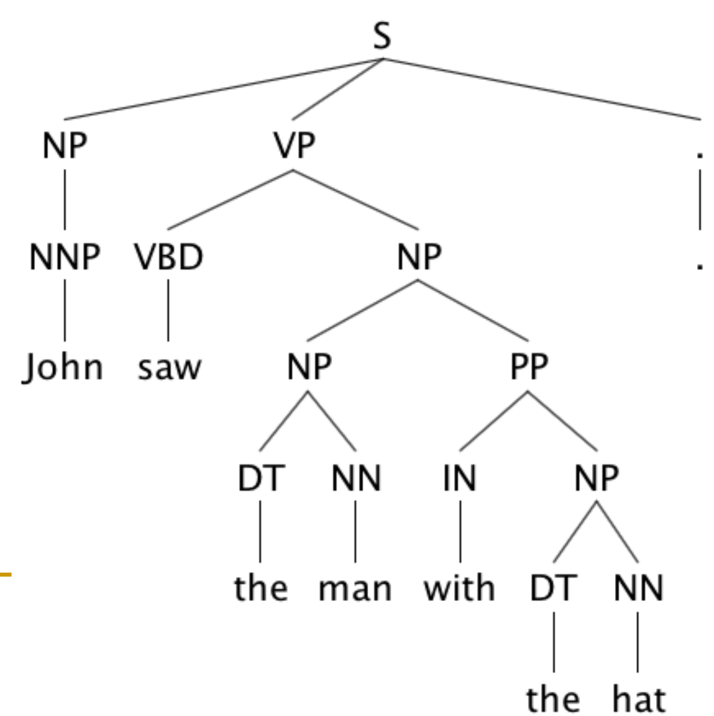
PCFGs: Example of lexical dependency

- choice 1: $VP \rightarrow VBD\ NP\ PP$ with P_1
- choice 2: $VP \rightarrow VBD\ NP$ with P_2
  $NP \rightarrow NP\ PP$ with P_3
- Choice depends on a word: *moon* vs *man*
- But PCFG will always prefer 1 or 2
  - depends on P_1, P_2, P_3
- Frequency of the rule is not enough!

John saw (*the* ***moon***) (*with the* ***telescope***)

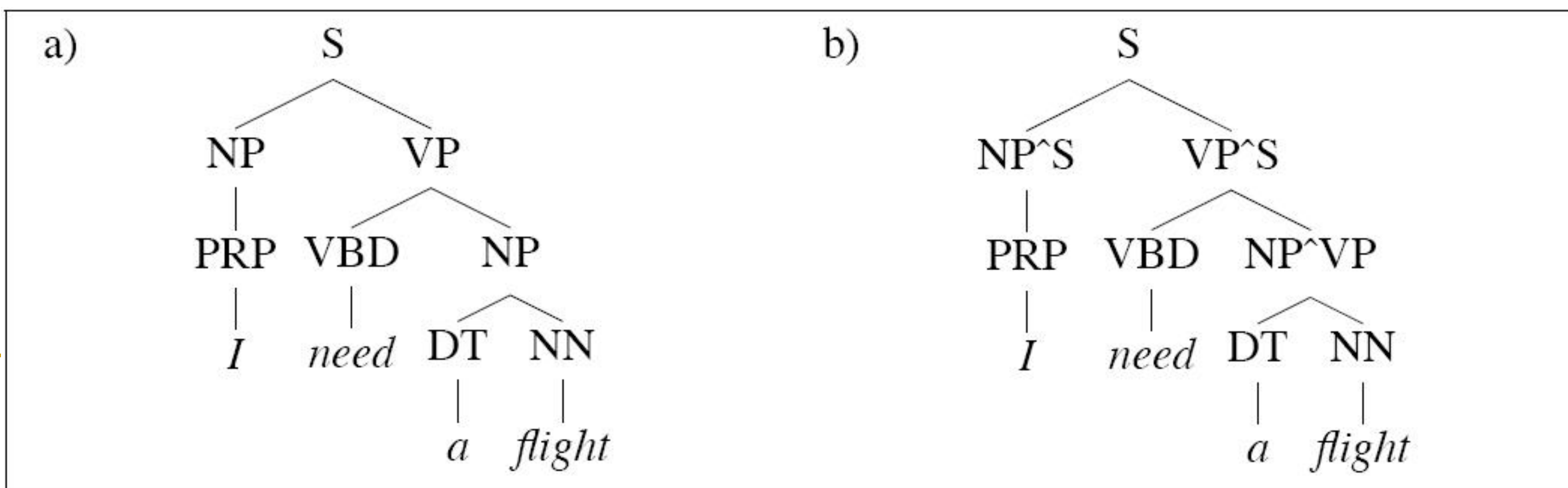


John saw (*the* ***man*** *with the* ***hat***)



Improving PCFSs by parent annotation

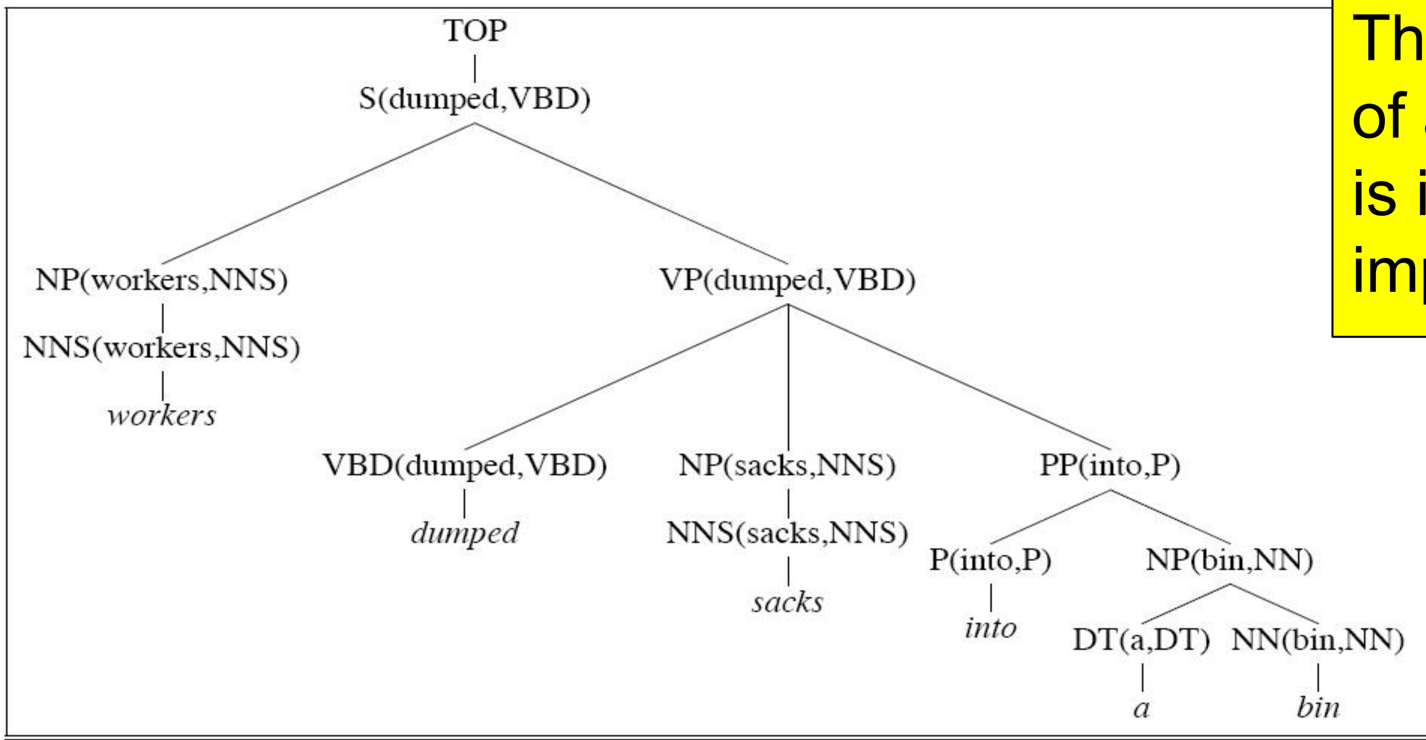
- For solving the *structural problem*, one could *split* the non-terminals into many versions
  - For the example: split NP in $NP_{subjects}$ and $NP_{direct\text{-}objects}$
- A way to implement it: parent annotation on the nodes
  E.g.: $VP \rightarrow VDB\ NP \quad p_1$
  $VP\hat{}S \rightarrow VDB\ NP\hat{}VP \quad p_2 \neq p_1$
- Use a grammar that parent-annotate the phrasal non-terminals
- Excluding or including pre-terminals (POS's)



Lexicalized parse trees

- For solving the *lexical problem*, each non-terminal symbol in parse tree is annotated with a single word (its *lexical head* or *headword*) and POS

The lexical head of a constituent is its "most important" word



Internal Rules					**Lexical Rules**		
TOP	→	S(dumped,VBD)			NNS(workers,NNS)	→	workers
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)		VBD(dumped,VBD)	→	dumped
NP(workers,NNS)	→	NNS(workers,NNS)			NNS(sacks,NNS)	→	sacks
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS)	PP(into,P)	P(into,P)	→	into
PP(into,P)	→	P(into,P)	NP(bin,NN)		DT(a,DT)	→	a
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)		NN(bin,NN)	→	bin

Lexicalized Probabilistic CFGs

- The headword for a node is set to the headword of its head daughter, and the head tag to the POS tag of the headword
  - E.g.: VP(dumped,VDB) → VDB(dumped,VDB) NP(sacks,NNS) PP(into,P)

$$p = \frac{C\left(\text{VP(dumped,VDB)} \rightarrow \text{VDB(dumped,VDB) NP(sacks,NNS) PP(into,P)}\right)}{C\left(\text{VP(dumped,VDB)}\right)}$$

- **lexical rules** express expansions of *preterminals* (i.e., POS's) to words
  - Deterministic (i.e., they have probability 1): a lexicalized preterminal like NN(bin,NN*)* can only expand to the word *bin*:

    NN(bin, NN) → *bin*   *P*=1
- **internal rules** express the other rule expansions
  - I.e.: *VP* → *NP PP*   *P*=?
  - For the internal rules we will need to estimate probabilities

Lexicalized Probabilistic CFGs

- The “head daughter” is the “most important” daughter of the node
  - Choosing such head daughters is complicated and indeed controversial
  - Determined by a set of rules defined in modern linguistic theories of syntax
    - See the Jurafsky’s book for an example
- Usually the “dead daughter”, and thus the lexical head, are just learned from a lexicalized treebank
- For calculating P
  - We need a **huge** corpus! Probably most of those rules would be associated with a zero probability

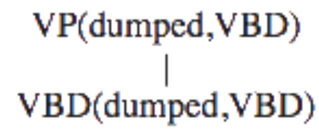
Lexicalized Probabilistic CFGs

- Make some independence assumptions:
  - break down each rule, so that we would estimate the probability as the product of smaller independent probability estimates…
  - …for which we could acquire reasonable counts
- Collins' model 1 (simplified)
  - A CFG rule is thought of as: LHS → L_n L_{n-1} … L_1 H R_1 … R_{n-1} R_n
  - Generative story:
    - First, generate the head of the rule (i.e., H)
    - Then, generate the dependents of the head, one by one, from the inside out
    - Each of these generation steps will have its own probability
    - The STOP non-terminal at the left and right edges of the rule…
      - …will allow the model to know when to stop generating dependents on a given side
  - E.g.: *VP*(*dumped,VBD*) → $\underbrace{\textit{STOP}}_{L_1}$ $\underbrace{\textit{VBD(dumped,VBD)}}_{H}$ $\underbrace{\textit{NP(sacks,NNS) PP(into,P)} \text{ STOP}}_{R_1\,R_2\,R_3}$

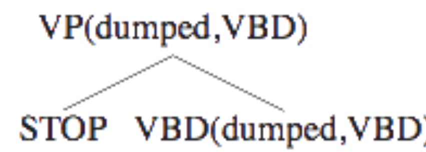
“generate” == search into the treebank

Collin’s Model 1 (simplified)

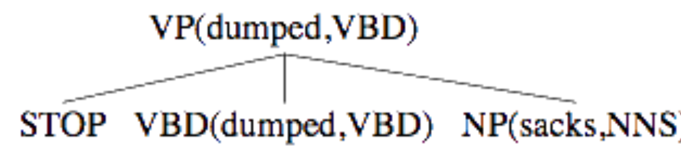
1) First generate the head VBD(dumped,VBD) with probability
$P_H(\text{H}|\text{LHS}) = \text{P(VBD(dumped,VBD)} \mid \text{VP(dumped,VBD))}$



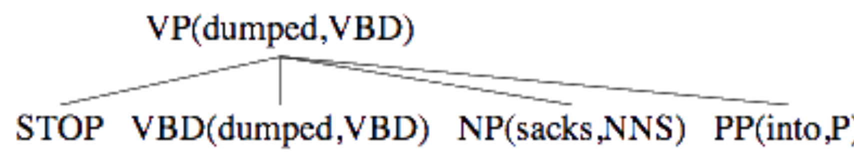
2) Then generate the left dependent (which is STOP, since there isn’t one) with probability
$P_L(\text{STOP}|\ \text{VP(dumped,VBD),VBD(dumped,VBD))}$



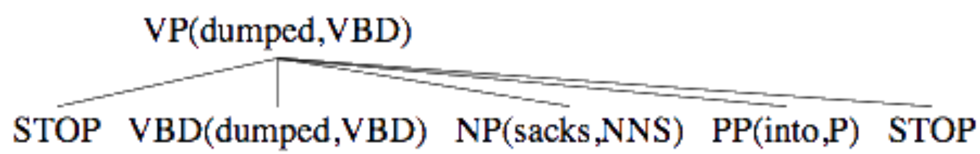
3) Then generate right dependent NP(sacks,NNS) with probability
$P_R(\text{NP(sacks,NNS}|\ \text{VP(dumped,VBD), VBD(dumped,VBD))}$



4) Then generate the right dependent PP(into,P) with probability
$P_R(\text{PP(into,P)} \mid \text{VP(dumped,VBD), VBD(dumped,VBD))}$



5) Finally generate the right dependent STOP with probability
$P_R(\text{STOP} \mid \text{VP(dumped,VBD), VBD(dumped,VBD))}$



Collin's Model 1 (simplified)

- Thus:

$$P(VP(dumped,VBD) \rightarrow VBD(dumped,VBD)\ NP(sacks,NNS)\ PP(into,P)) =$$
$$= P_H(VBD(dumped,VDB) \mid VP(dumped,VDB)) \times$$
$$P_L(STOP \mid VP(dumped,VDB), VBD(dumped,VDB)) \times$$
$$P_R(NP(sacks,NNS) \mid VP(dumped,VDB), VBD(dumped,VDB)) \times$$
$$P_R(PP(into,P) \mid VP(dumped,VBD), VDB(dumped,VDB)) \times$$
$$P_R(STOP \mid VP(dumped,VDB), VBD(dumped,VDB))$$

- And, for example:

$$P_R(NP(sacks,NNS) \mid VP(dumped,VDB), VDB(dumped,VDB)) =$$
$$= \frac{C\left(VP(dumped,VDB) \rightarrow \text{...<everything>...}\ VDB(dumped,VDB)\ NP(sacks,NNS)\ \text{...<everything>...}\right)}{C\left(\text{VP(dumped,VDB)}\right)}$$

$$P_L(STOP \mid VP(dumped,VDB), VDB(dumped,VDB)) =$$
$$= \frac{C\left(VP(dumped,VDB) \rightarrow VDB(dumped,VDB)\ \text{...<everything>...}\right)}{C\left(\text{VP(dumped,VDB)}\right)}$$

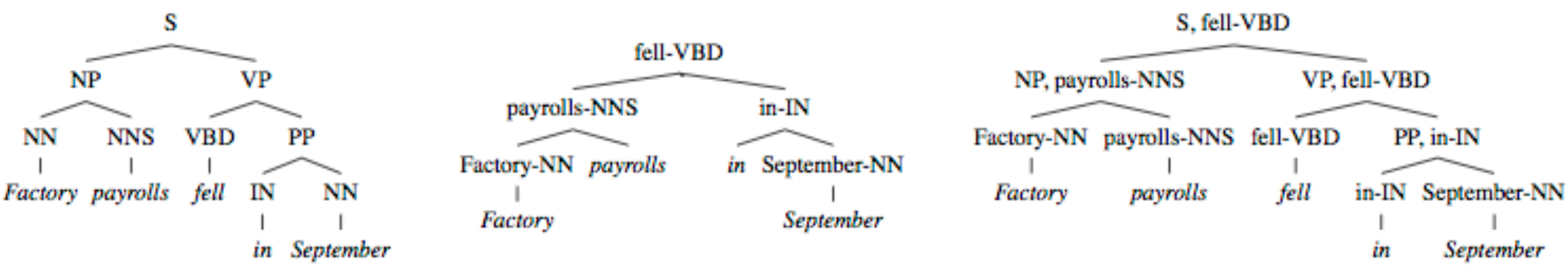
$$P_H(VDB(dumped,VDB) | VP(dumped,VDB)) =$$
$$= \frac{C(VP(dumped,VBD) \rightarrow\ \ldots < \text{everyth.} >\ \ldots\ \ VBD(dumped,VBD)\ \ldots < \text{everyth.} >\ \ldots)}{C\left(VP(dumped,VBD)\right)}$$

Stanford Parser

- Lexicalized PCFG are hard to train
  - Data sparsity
- Lexicalized parse tree can be seen as a combination of:
  - Unlexicalized parse tree
  - Dependency tree among words
- These trees can be trained separately
  - Reduce data sparsity

Stanford Parser

- Starting form a lexicalized Treebank:
  - Reconstruct unlexicalized PCFG
  - Extract headwords and build dependencies among them
- Parsing:
  - P(T): probability of a given unlexicalized parse tree
  - p(D): probability of a given dependency tree (sort of)
- Factored Lexicalized PCFG: p(T,D) = p(T)•p(D)



(a) PCFG Structure (b) Dependency Structure (c) Combined Structure

Demo

Charniak parser

- Based on a Lexicalized PCFG
  - Returns the 50 most probable parse trees
- A Maximum Entropy models reranks the trees and selects the best one
  - 13 feature template, predicating on parse trees
  - A large number of feature instances!

Corpora

- Corpus: a collection of tagged texts
  - Human experts tag texts, by hand, setting the so-called *gold standard*
  - Used to train statistic models
- Well-known corpora, among others:
  - POS tags: Brown Corpus
  - Chunk: CoNLL
  - Parsing: Penn Treebank

How corpora are built

- Bootstrap
  1. Tag by hand a subset of the corpus
  2. Train a model
  3. Use the model to tag a larger subset of the corpus
  4. Revise and fix tagging
  5. Go to 2
- Kappa measure: agreement among human taggers
  - Human taggers do not fully agree, usually
  - We need a measure of such agreement

Kappa measure

- Compute agreement as:

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

$P(A)$ is the observed agreement among the raters

$P(E)$ is the expected agreement (i.e., $P(E)$ is the probability that raters agree by chance)

- The values of the agreement:

1: "perfect agreement"
0: "agreement is equal to chance"
<0: "agreement worse than chance"

- Many formulas for calculating $P(A)$ and $P(E)$
  - Cohen's kappa, Scott's Pi, Fleiss' kappa, Randolph's K_{free}, …

Dependency parsing

Other syntactic models

- Grammatical relationships
  - Subject
    - I booked a flight to New York
    - The flight was booked by my agent.
  - Direct object
    - I booked a flight to New York
  - Complement
    - I said that I wanted to leave

Dependency Grammars

- In CFG-style phrase-structure grammars the main focus is on *constituents.*
- But it turns out you can get a lot done with just binary relations among the words in an utterance.
- In a dependency grammar, a parse is a tree where
  - the nodes stand for the words in an utterance
  - The links between the words represent dependency relations between pairs of words.
    - Relations may be typed (labeled), or not.
- Approach based on European tradition (from ancient Greece); not so used in America

Dependency Grammars

- Suited for languages having many variations in word ordering
- Jarvinen & Tapanaien 97
- Link Grammar (Sleator & Temperley 93)
- Constraint Grammar (Karlsson et al. 95)

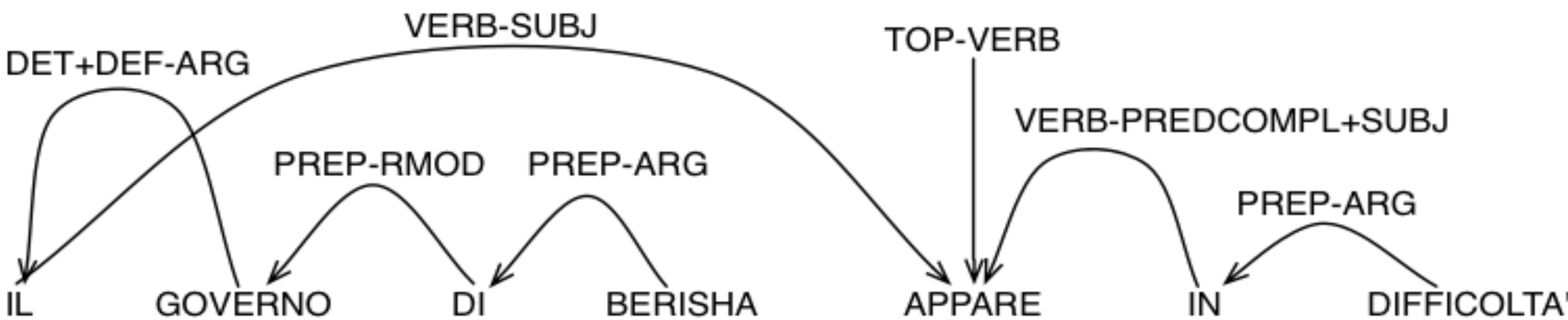
Dependencies

Dependency	Description
Subj	syntactic subject
Obj	direct object (incl. Sent. Compl.)
Dat	indirect object
Pcomp	complement of a preposition
Comp	predicate nominals (copulas' compl)
Tmp	temporal adverbials
Loc	location adverbials
Attr	premodifying (attributive) nominals (gen., etc.)
Mod	nominal postmodifiers (prepos. phrases, etc.)

Dependency Grammar: TUT

- TUT Treebank contains DG-tagged sentences

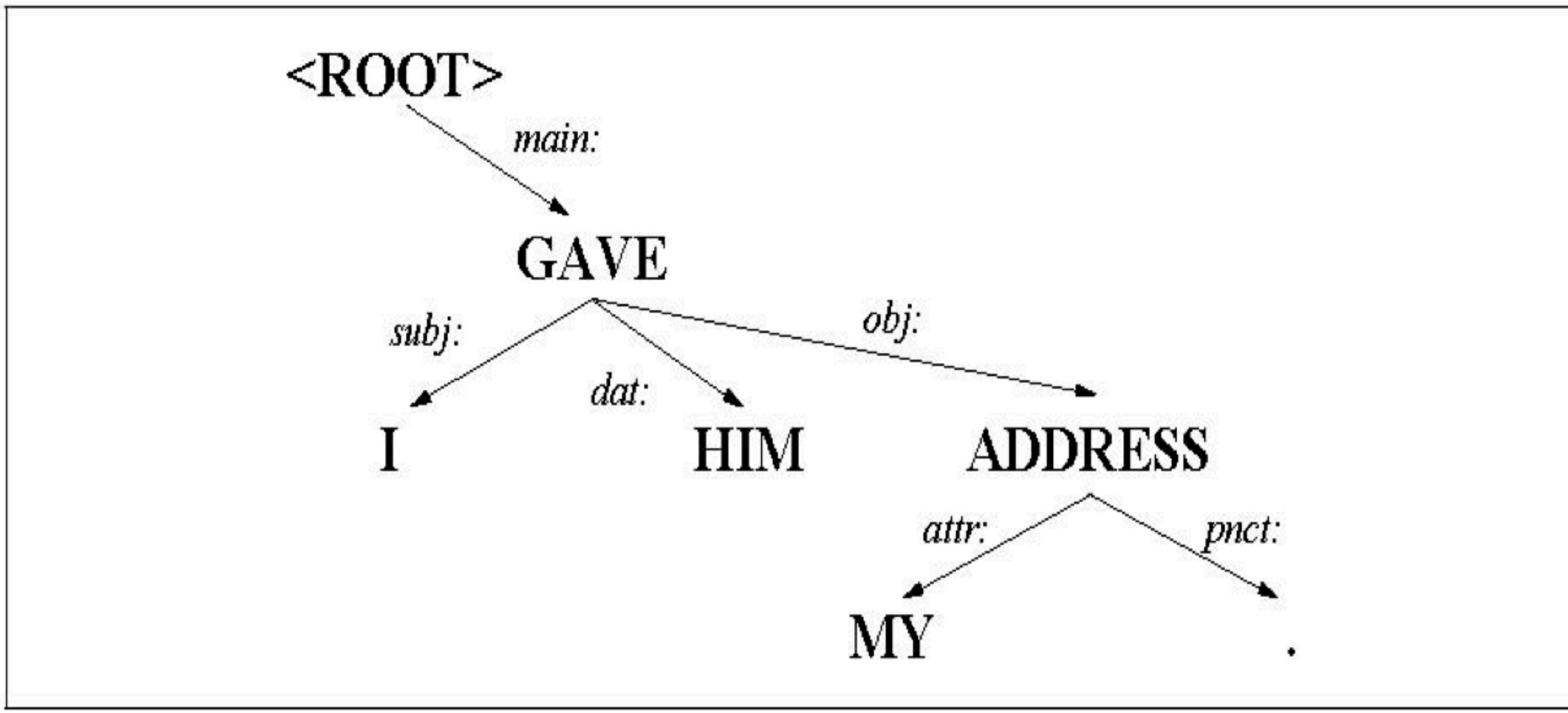
1 Il (IL ART DEF M SING) **[5;VERB-SUBJ]**
2 Governo (GOVERNO NOUN COMMON M SING) **[1;DET+DEF-ARG]**
3 di (DI PREP MONO) **[2;PREP-RMOD]**
4 Berisha (BERISHA NOUN PROPER) **[3;PREP-ARG]**
5 appare (APPARIRE VERB MAIN IND PRES INTRANS 3 SING) **[0;TOP-VERB]**
6 in (IN PREP MONO) **[5;VERB-PREDCOMPL+SUBJ]**
7 difficolta' (DIFFICOLTÀ NOUN COMMON F ALLVAL) **[6;PREP-ARG]**
8 . (#\. PUNCT) **[5;END]**



Dependency Parsing

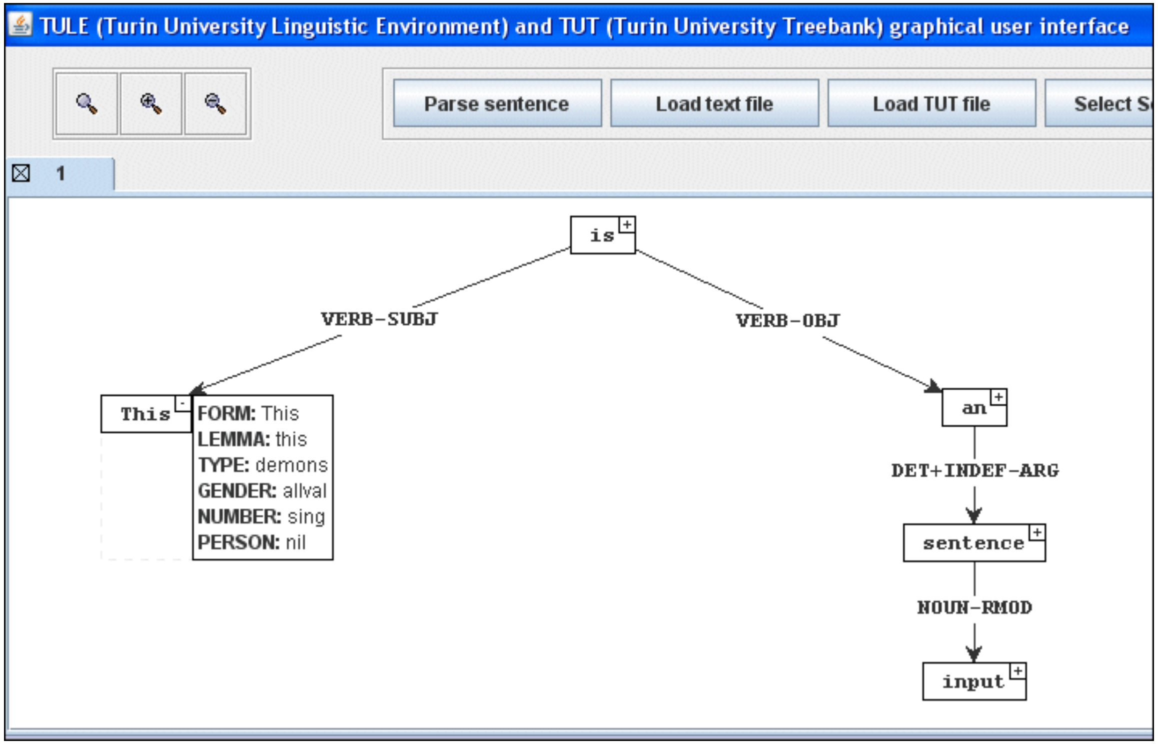
- Links from word to word, instead of constituent units
- Approach based on European tradition (from ancient Greece); not so used in America
- The *Subject* and *Object* concepts are precursors of subcategorization (also known as 'valence') and linked to the Dependency theory (Dependency Grammar)
- Dependency parsing is widely used as a computational model
- The analysis of relationships among words is useful at several levels

Dependency Parsing



TULE

- Dependency grammar-based (TUT)
- Client-server
  - Server: LISP-based parser
  - Client: Java-based GUI
- Multilanguage



REFERENCES

Full parsers

- Stanford parser
  - http://nlp.stanford.edu/software/lex-parser.shtml
- Charniak parser
  - http://www.cs.brown.edu/~ec/
- NLTK (actually, contains a lot of tools…)
  - http://www.nltk.org
- TULE
  - http://www.tule.di.unito.it/

Corpora/1

- Linguistic Data Consortium
  - http://www.ldc.upenn.edu
- European Language Resources Association (ELRA)
  - http://www.icp.grenet.fr/ELRA/
- Int. Computer Archive of Modern English (ICAME)
  - http://nora.hd.uib.no/icame.html
- Oxford Text Archive (OTA)
  - http://ota.ahds.ac.uk/
- Child Language Data Exchange System (CHILDES)
  - http://childes.psy.cmu.edu/

Corpora/2

- NLTK_lite (directory **`corpora`**)
  - Small samples of: Penn Treebank, Brown, ecc.
- Penn Treebank:
  - http://www.cis.upenn.edu/~treebank/
- Brown Corpus:
  - http://en.wikipedia.org/wiki/Brown_Corpus
- American National Corpus:
  - http://americannationalcorpus.org/
- British National Corpus:
  - http://www.natcorp.ox.ac.uk/
- Corpus e Lessico di Frequenza dell'Italiano Scritto:
  - http://alphalinguistica.sns.it/CoLFIS/CoLFIS_Presentazione.htm