Natural Language Processing II (SC674)

Prof. Feng Zhiwei

Ch5. Lexicalized and Probabilistic Parsing

5.1 Probabilistic Context-Free grammar (PCFG)

5.1.1 The definition of PCFG:

The Probabilistic Context-Free Grammar (PCFG) also known as the Stochastic Context-Free Grammar (SCFG).

A context-free grammar G is defined by four parameters {N, Σ, P, S}:

- A set of non-terminal symbols N;
- A set of terminal symbols Σ (disjoint from N);
- A set of productions P, each of the form A → β, where A is a non-terminal symbol and β is a string of symbols from the infinite set of strings (Σ ∪ N);
- A designated start symbol S.

A PCFG augments each rule in p with a conditional probability:

$$A \rightarrow \beta \ [p]$$

A PCFG is thus a 5-tuple G = {N, Σ, P, S, D}, where D is a function assigning probabilities to each rule in P. This function expresses the probability p that the given non-terminal A will be expanded to the sequence β; it is often referred to as

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

or as

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

Formally this is conditional probability of a given expansion given the left-hand-side non-terminal A. If we consider all the possible expansions of a non-terminal, the sum of their probabilities must be 1.

For example, the rules of PCFG of our small CFG in Chapter 3 can be:

S → NP VP	[.80]
S → AUX NP VP	[.15]
S → VP	[.05]
NP → Det Nominal	[.20]
NP → Proper Noun	[.35]
NP → Nominal	[.05]
NP → Pronoun	[.40]
Nominal → Noun	[.75]
Nominal → Noun Nominal	[.20]
Nominal → Proper-Noun Nominal	[.05]
VP → Verb	[.55]
VP → Verb NP	[.40]
VP → Verb NP NP	[.05]

Det → that [.05] | this [.80] | a [.15]

Noun → book [.10] | flight [.50] | meat [.40]

Verb → book [.30] | include [.30] | want [.40]

Aux → can [.40] | does [.30] | do [.30]
Proper Noun → Houston [.10] | ASIANA [.20] | KOREAN AIR [.30] | CAAC [.25] | Dragon Air [.15]
Pronoun → you [.40] | I [.60]

These probabilities are not based on a corpus; they were made up merely for expository purpose.

Note that the probabilities of all of the expansion of a non-terminal sum to 1.

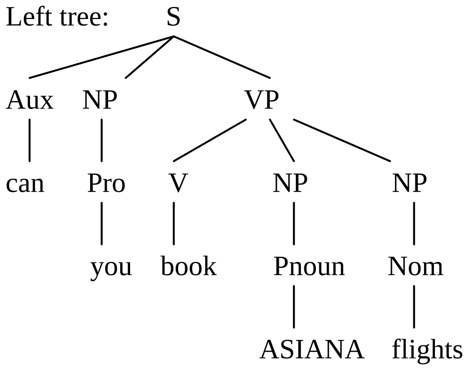
A PCFG assigns a probability to each parse-tree T of a sentence S. The probability of a particular parse T is defined as the product of the probabilities of all the rules r used to expand each node n in the parse tree:

$$P(T) = \prod_{n \in T} p(r(n))$$

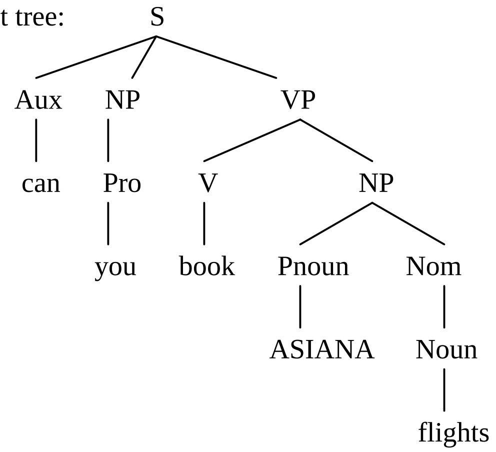
For example, the sentence "Can you book ASIANA flights" is ambiguous:

One meaning is :"Can you book flights on behalf of ASIANA", the other meaning is : "Can you book flights run by ASIANA". The trees are respectively as follows:

Left tree:



Right tree:



The probabilities of each rule in left tree are:

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 → ASIANA	[.40]
Noun → flights	[.50]

The probabilities of each rule in right tree are:

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 → ASIANA	[.40]
Noun → flights	[.50]

The P (T_l) of left tree is:

$$P(T_l) = .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 * .30 * .40 * .50$$
$$= 1.5 \text{ X } 10^{-6}$$

The P (T_r) of right tree is:

$$P(T_l) = .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 * .30 * .40 * .50$$
$$= 1.7 \text{ X } 10^{-6}$$

We can see that the right tree has a higher probability. Thus this parse will be chosen as correct result.

The disambiguation algorithm picks the best tree for a sentence S out of the set of parse trees for S (which we shall call τ (S). We want the parse tree T which is most likely given the sentence S. Formally, if $T \in \tau(S)$, the tree with the highest probability T(S) will equal to " argmax P(T)", so we have

$$T(S) = \text{argmax } P(T)$$

5.1.2 Probabilistic CYK algorithm

How we can use PCFG in the parsing algorithm? Luckily, the algorithm for computing the most-likely parse are simple extensions of the standard algorithms for parsing.

Now we shall present the **probabilistic CYK** (Cocke-Younger-Kasami) **algorithm**.

The Probabilistic CYK parsing algorithm was first described by Ney (1991).

Assume that the PCFG is in CNF (Normal Chomsky Form). The grammar is in CNF if it is ε -free and if in addition each production is either of the form A → BC or A → α . In Chapter 3, we introduced the CYK algorithm. Now we shall give the formal description of Probabilistic CYK algorithm.

Formally, the probabilistic CYK algorithm assume the following input, output, and data structure:

- **Input**.
  1. A Chomsky normal form PCFG $G = \{N, \Sigma, P, S, D\}$.Assume that the $|N|$ non-terminals have indices $1, 2, \ldots, |N|$, and that the start symbol S has index 1.
  2. n words $w_1 \ldots w_n$.
- .**Data structure**.

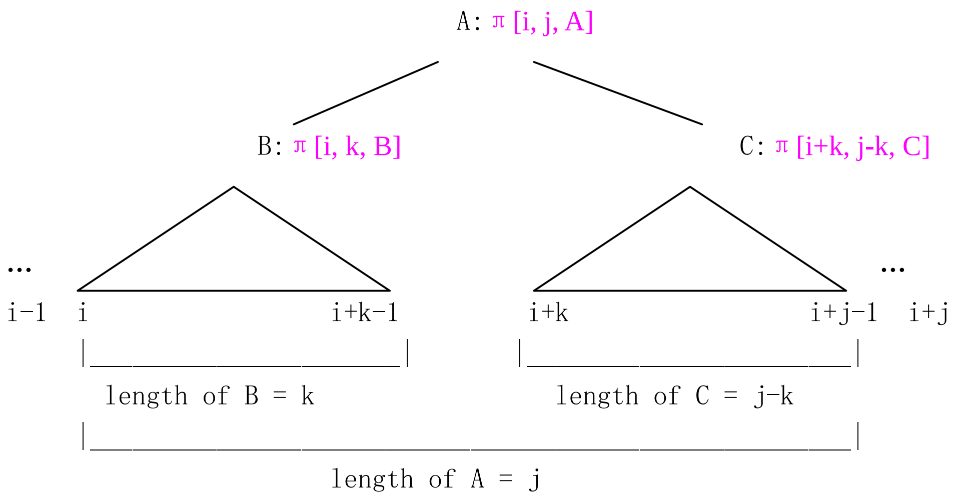
  A dynamic programming array π [i, j, a] holds the maximum probability for a constituent with non-terminal index a spanning words i … j.

- **Output**.

  The maximum probability parse will be π [1, n, S]: The parse tree whose root is S and which spans the entire string of words $w_1 \dots w_n$.

The CYK algorithm fills out the probability array by induction. Here we use w_{ij} to mean the string of words from word i to word j. The induction is as following:

- **Base case**: Consider the input strings of length one (individual words w_i). In CNF, the probability of a given non-terminal A expanding to a single word w_i must come only from the rule A → w_i.
- **Recursive case**: For string of words of length >1, A => w_{ij} if and only if there is at least one rule A → BC and some k, $1 \leqslant k < j$, such that B derives the first k symbols of w_{ij} and C derives the last j – k symbols of w_{ij}, their probability will already be stored in the matrix π. We compute the probability of w_{ij} by multiplying together the probability of these two pieces. But there may be multiple parses of w_{ij}, and so we'll need to take max over all the possible divisions of w_{ij} (over all values of k and over all possible rules).



In the rule A → BC, the probability of A (w_{ij}) equals the product of the probability of B and C. such we can compute the probability P(T) for different sub-tree.

Then we can compute the highest probability T(S) of the sentence S. It will equal to argmax P(T):

$$T(S) = \operatorname{argmax} P(T)$$

5.1.3 Learning PCFG Probabilities

The PCFG probabilities come from corpus of already-parsed sentences: tree-bank.

Penn Tree-bank contains parse trees for the Brown Corpus, one million words from the Wall Street Journal, and parts of the Switchboard corpus.

Given a tree-bank, the probability of each expansion of a non-terminal can be computed by counting the number of times that expansion occurs and then normalizing:

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{Count}(\alpha \to \beta)}{\sum_{\gamma} \text{Count}(\alpha \to \gamma)} = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

“$\sum_{\gamma}$Count (α → γ)” is the rule number taking α as the LHS of the rule.

When a tree-bank is unavailable, the count needed for computing PCFG probabilities can be generated by first parsing a corpus.

- If sentences were unambiguous, it would be very simple: parse the corpus, add a counter for every rule in the parse, and then normalize to get probabilities.
- If the sentences were ambiguous, we need to keep a separate count for each parse of a sentence and weight each partial count by the probability of the parse it appears in.

5.2 Lexicalized PCFG

5.2.1 problems with PCFG:

- The problems in structural dependency

A CFG assumes that the expansion of any one non-terminal is independent of the expansion of any other non-terminal. This independence assumption is carried over in the PCFG: each PCFG rule is assumed to be independent of each other rule, and thus the rule probabilities are multiplied together. But, In English, the choice of how a node expands is dependent on the location of the node in the parse tree. For example, there is a strong tendency for the syntactic subject of a sentence to be a pronoun. This tendency is caused by the use of subject position to realize the topic or old information. Pronouns are a way to talk about old information. While the non-pronominal lexical noun-phrase are often used to introduce new referents. According to the investigation of Francis (1999), the 31,021 subjects of declarative sentences in Switchboard corpus, 91% are pronouns and only 9% are lexical. By contrast, out of 7,498 direct object, only 34% are pronoun, and 66% are lexical.

Subject: **She** is able to take her baby to work with her.
: **My wife** worked until we had a family.
Object: Some laws absolutely prohibit **it**.
All the people signed **applications**.

These dependencies could be captured if the probability of expanding an NP as a pronoun (via the rule NP → Pronoun) versus a lexical NP (via the rule NP → Det Noun) were dependent on whether the NP was a subject or an object. However, this is just the kind of probabilistic dependency that a PCFG does not allow.

- The problems in lexical dependency

PCFG can only be represented via the probability of pre-terminal nodes to be expanded lexically. But there are a number of other kinds of lexical and other dependencies that is important in modeling syntactic probabilities.

--PP-attachment: The lexical information plays an important role in selecting the correct parsing of an ambiguous prepositional phrase attachment.

For example, in the sentence “Washington sent more than 10,000 soldiers into Afghanistan”, PP “into Afghanistan” can be attached either to NP (more than 10,000 soldiers), or to attached to the verb (sent).

In PCFG, the attachment choice comes down to the choice between two rules:

NP → NP PP (NP-attachment)
And VP → VP PP (VP-attachment)

The probability of these two rules depends on the training corpus.

Corpus	NP-attachment	VP-attachment
AP Newswire (13 million words)	67%	33%
Wall Street Journal & IBM manuals	52%	48%

Whether the preference is 67% or 52%, in PCFG, this preference is purely structural and must be the same to all verb.

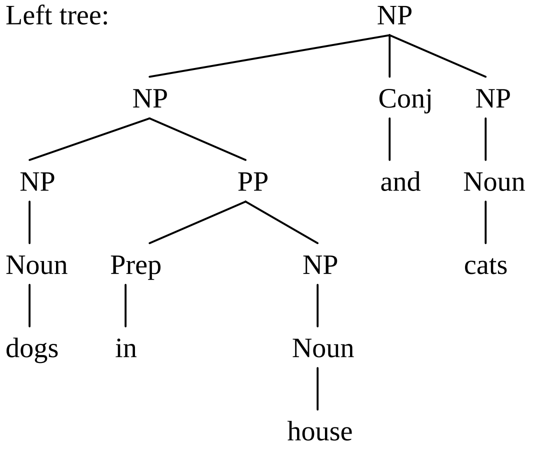
However, the correct attachment is to verb. The verb “send” subcategorizes for a destination, which can be expressed with the preposition “into”. It is a lexical dependency. The PCFG can not deal with the lexical dependency.

--Coordination ambiguities:

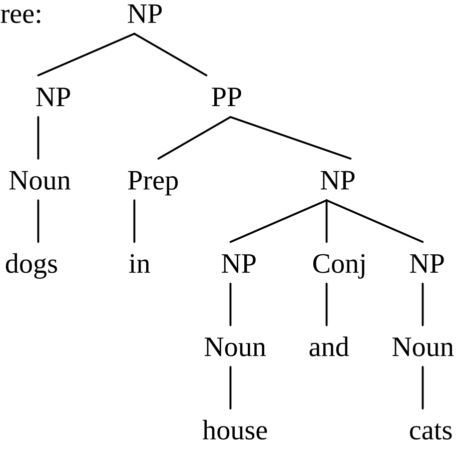
The coordination ambiguities are the key to choosing the proper parse.

In the phrase “dogs in houses and cats” is ambiguous:

Left tree:



Right tree:



Although the left tree is intuitively the correct one. But the PCFG will assign them identically probabilities because both structure use the exact same rule:

NP → NP Conj NP
NP → NP PP
NP → Noun
PP → Prep NP
Noun → dogs | house | cats
Prep → in
Conj → and

In this case, PCFG will assign two trees the same probability.

PCFG has a number of inadequacies as a probabilistic model of syntax, we shall augment PCFG to deal with these problems.

5.2.2 Probabilistic Lexicalized CFG

Charniak (1997) proposed the approach of the probabilistic representation of lexical heads. It is a kind of lexical grammar. In this probabilistic representation, each non-terminal in a parse-tree is annotated with a single word which is its lexical-head.

E,g. "Workers dumped sacks into a bin" can be represented as follows:

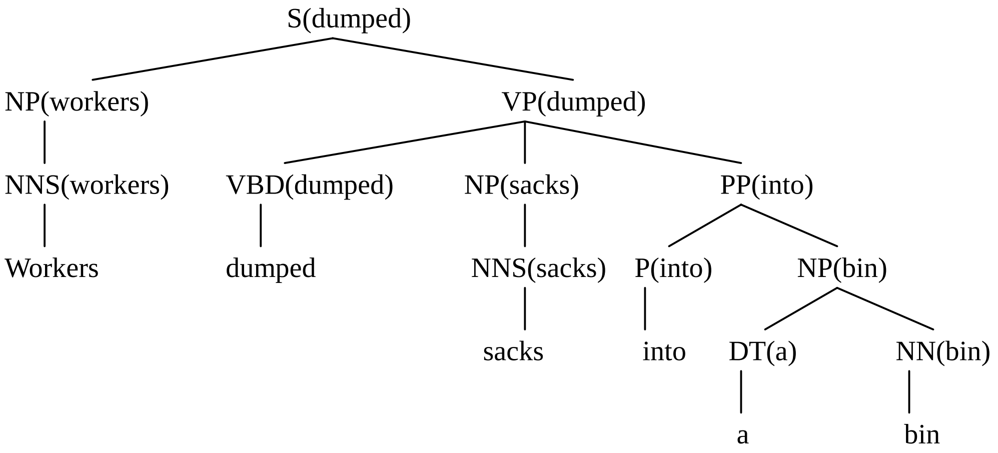


Fig. Lexicalized tree

In this case, we were to treat a probabilistic lexicalized CFG like a normal but huge PCFG. Then we would store a probability for each rule/head combination. E.g.

VP(dumped) → VBD(dumped) NP(sacks) PP(into) [3X10^{-10}]
VP(dumped) → VBD(dumped) NP(cats) PP(into) [8X10^{-11}]
VP(dumped) → VBD(dumped) NP(hats) PP(into) [4X10^{-10}]
VP(dumped) → VBD(dumped) NP(sacks) PP(above) [1X10^{-12}]

This sentence can be also parsed to another tree:

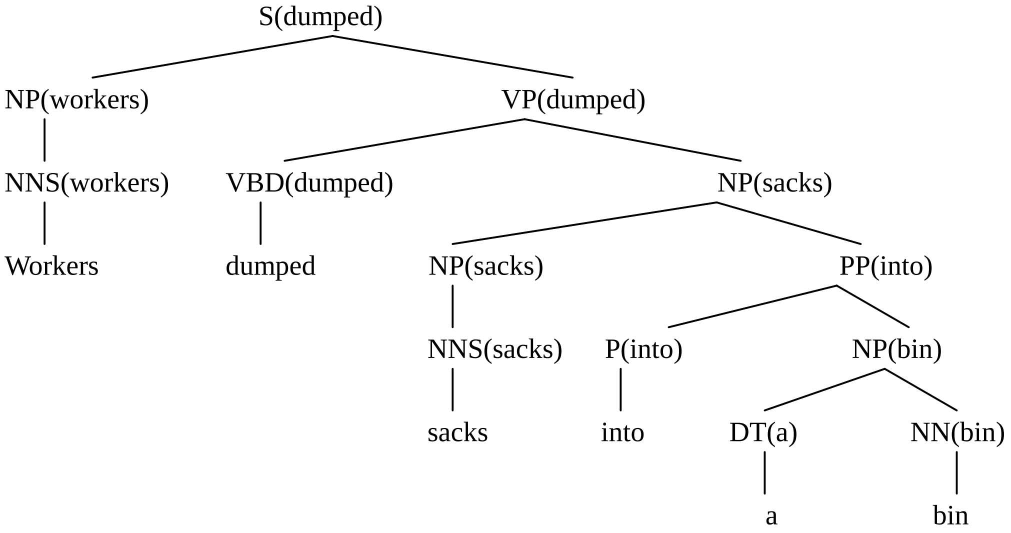


Fig. Incorrect parse-tree

If VP(dumped) expands tp VBD NP PP, then the tree will be correct. If VP(dumped) expands to VBD NP, then the tree will be incorrect.

Let us compute both of these by counting in the Brown Corpus portion of the Penn Trre-bank. The first rule is quite likely:

$$P(VP \rightarrow VBD\ NP\ PP|VP,\ dumped) = \frac{C(VP(dumped) \rightarrow VBD\ NP\ PP)}{\sum_{\beta} C(VP(dumped) \rightarrow \ \beta}$$

$$= 6/9 = .67$$

The second rule never happens in the Brown Corpus. This is not surprising, since “dump” is a verb of caused-motion into a new location:

$$P(VP \rightarrow VBD\ NP|VP,\ dumped) = \frac{C(VP(dumped) \rightarrow VBD\ NP)}{\sum_{\beta} C(VP(dumped) \rightarrow \ \beta}$$

$$= 0/9 = 0.$$

In practice this zero value would be smoothed somehow, but now we just notice that the first rule is preferred.

For the head probabilities we can also count it using same method.

In the correct parse, a PP node whose mother’s head (X) is “dumped’ has the head ‘into”. In the incorrect parse, a PP node whose mother’s head (X) is “sacks’ has the head “into”. We can use counts from Brown portion of the Penn Tree-bank. X is the mother’s head.

$$P(into\ |PP,\ \ dumped) = \frac{C(X(dumped) \rightarrow \ldots PP\ (into)\ldots)}{\sum_{\beta} C(X(dumped) \rightarrow \ldots PP\ldots)}$$

$$= 2/9 = .22$$

$$P(into\ |PP,\ \ sacks) = \frac{C(X(sacks) \rightarrow \ldots PP\ (into)\ldots)}{\sum_{\beta} C(X(sacks) \rightarrow \ldots PP\ldots)}$$

$$= 0/0 = ?$$

Once again, the head probabilities correctly predict that “dumped” is more likely to be modified by “into” than is “sacks”.

5.3 Human Parsing

In the last 20 years we have learned a lot about human parsing. Here we shall give a brief overview of some recent results.

5.3.1 Ambiguity solution in the human parsing:

Human sentence processor is sensitive to **lexical sub-categorization preferences**. For example, The scientists can ask the people to read the ambiguous sentence and check off a box indicating which of the two interpretations they got first. The results are after each sentence.

“The women kept the dogs on the beach”

- The women kept the dogs which were on the beach. 5%
- The women kept them (the dogs) on the beach. 95%

“The women discussed the dogs on the beach”

- The women discussed the dogs which were on the beach. 90%

- The women discussed them (the dogs) while on the beach. 10%

The results were that people preferred VP-attachment with "keep" and NP-attachment with "discuss".

This suggest that "keep" has a sub-categorization preference for a VP with three constituents: (VP → V NP PP) while "discuss" has a sub-categorization preference for a VP with to constituents: (VP → V NP), although both verbs still allow both sub-categorizations.

5.3.2 Garden-path sentences:

The garden-path sentence is a specific class of temporarily ambiguous sentences.

For example, "The horse raced past the barn fell" ("barn" is a farm building for storing corps and food for animals). .

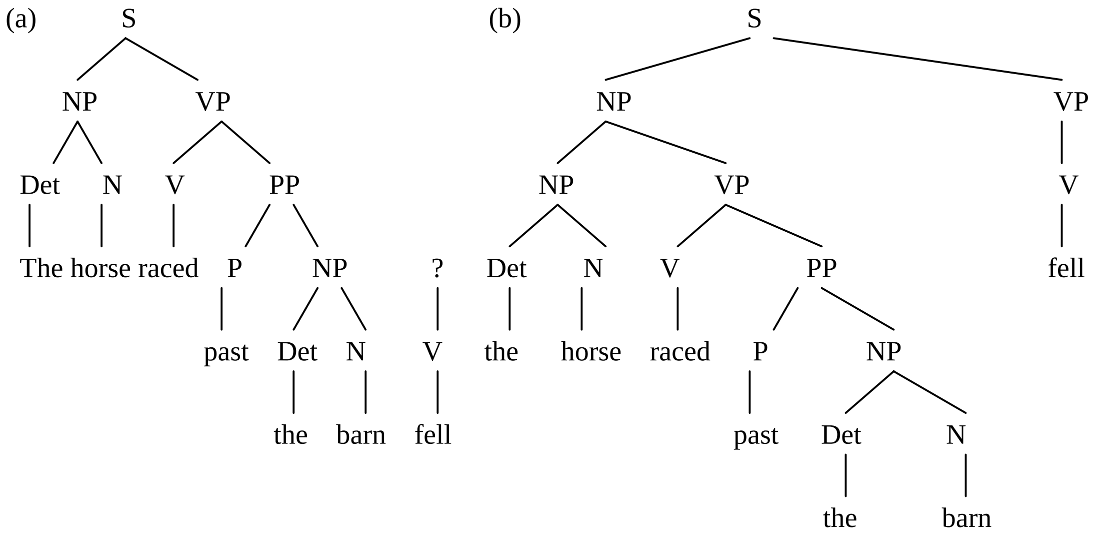


Fig. Garden-path sentence 1

The garden-path sentences are the sentences which are cleverly constructed to have three properties that combine to make them very difficult for people to parse:

- They are temporarily ambiguous: the sentence is not ambiguous, but its initial portion is ambiguous.
- One of the two or three parses in the initial portion is somehow preferable to the human parsing mechanism.
- But the dispreferred parse is the correct one for the sentence.

The result of these three properties is that people are "led down the garden path" toward the incorrect parse, and then are confused when they realize it is the wrong way.

More examples

"The complex houses married and single students and their families."

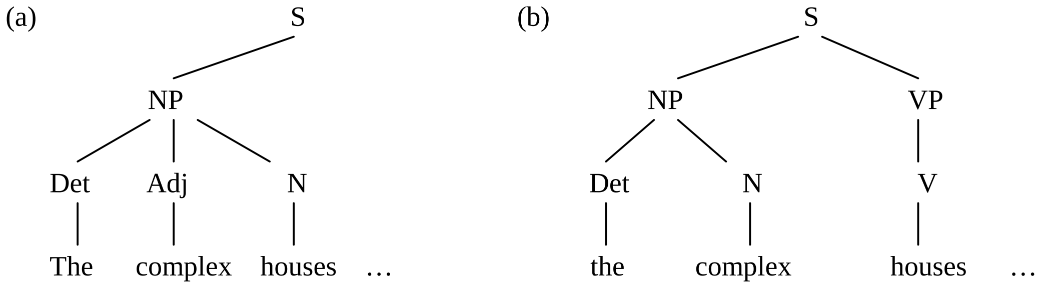


Fig. Garden-path sentence 2

In this sentence, the readers often mis-parse "complex" as Adj and "houses" as N, but the correct parse is to parse “complex’ as N and ‘houses’ as V, even it is dispreferable.

“The student forgot the solution was in the back of the book.”

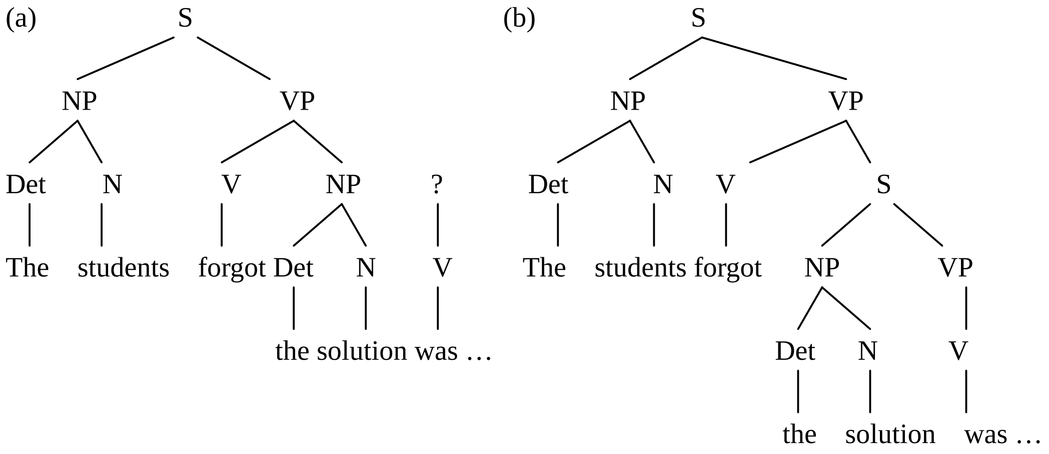


Fig. Garden-path sentence 3

In this sentence, the readers often mis-parse “the solution” as the direct object of “forgot” rather than as the subject of an embedded sentence. This is another sub-categorization preference difference: “forgot’ prefers a direct object (VP → V NP) to a sentential complement (VP → V S). The garden-path sentence is caused by the sub-categorization preferences of the verb.

In the sentence “The horse raced past the barn fell”, verb “raced” is preferable to be used as Main Verb (MV), and it dispreferable to be used as Reduced-Relative (RR). But correct one just the dispreferable (RR). MV/RR = 387

In the sentence “The horse found in the barn died”, since the verb “found” is transitive, the reduced-relative (RR) interpretation becomes much more than it was for “raced”. Its MV/RR probabilistic ratio (lower than 5) is less than MV/RR probabilistic ratio of “raced” (387). So this sentence cannot become the garden-path sentence.

The MV/RR probability ratio for “raced’ much more than the MV/RR probabilistic ratio for “found”. Perhaps, It is the explanation for garden-path sentence.

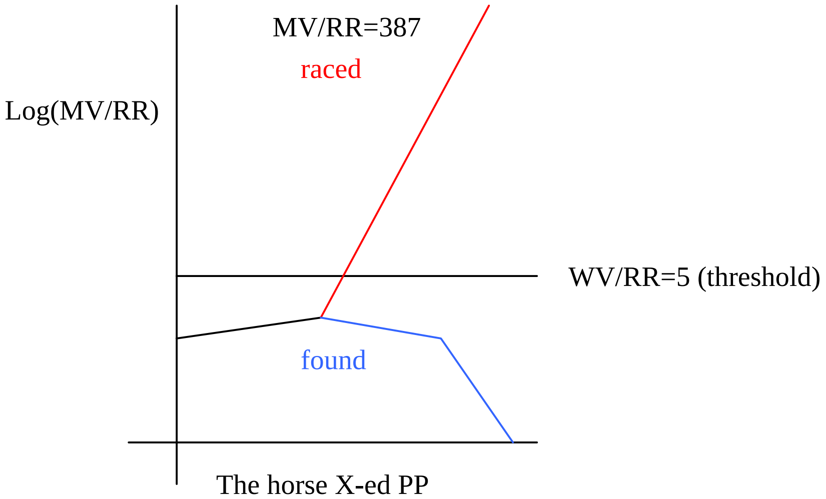


Fig. MV/RR probabilistic ratio

The model assumes that people are unable to maintain very many interpretation at one time.

Whether because of memory limitations, or just because they have a strong desire to come up with a simple interpretation, they prune away low-ranking interpretation. An interpretation is pruned if its probability is 5 times lower than the most-probable interpretation. The result is that they sometimes prune away the correct interpretation. Leaving a highest but incorrect interpretation. This is what happens with the probability (but "correct") reduced-relative (RR) interpretation in the sentence "The horse raced past the barn fell".

In above Figure, the WV/RR probabilistic ratio for "raced" falls above the threshold and the RR interpretation is pruned. For "found" its interpretation is active in the disambiguating region.