CSME 206A – Natural language & Speech Processing **Spring Semester**

Lecture 9: Syntax Analysis

Lecturer: K.R. Chowdhary *: Professor of CS*

9.1 Introduction

Syntax is the study of the principles and processes by which sentences are constructed in a particular languages. Syntactic investigation of a given language has as its goal the construction of a grammar that can be viewed as a device of some sort for producing the sentences of the language under analysis. More generally, linguists must be concerned with the problem of determining the fundamental underlying properties of successful grammars.

The central notion in linguistic theory is that of "linguistic level." A linguistic level, such as phonemics, morphology, phrase structure, is essentially a set of descriptive devices that are made available for the construction of grammars; it constitutes a certain method for representing utterances. We can determine the adequacy of a linguistic theory by developing rigorously and precisely the form of grammar corresponding to the set of levels contained within this theory, and then investigating the possibility of constructing simple and revealing grammars of this form for natural languages [noamch57].

9.2 Generative Methodology

Noam Chomsky published Syntactic Structures in 1957 ushering in the generative era of linguistic theory [noamcho59]. The essential paradigm shift or methodological innovation was that linguistic analysis was no longer an entirely 'bottom-up', data-driven purely empirical process, but rather, generative linguists started out with a meta-theory of what grammars of human languages look like and attempted to express specific grammars within this meta-theory. Such grammars are generative because they consist of finite sets of rules which should predict all and only the infinite grammatical sentences of a given human language (and what is conveyed about their meaning by their grammatical structure). Thus generative grammars define well-formed sets or mappings between sentences and (part of their) meanings.

Generative grammar got going at much the same time that theoretical computer science, and much of the theory of parsing and compiling programming languages has its antecedents in early generative linguistics (The Chomsky Hierarchy, etc.). For example context-free grammars and Backus-Naur notation are weakly equivalent formalisms, generating the same class of context-free languages, which seem quite appropriate for capturing the hierarchical structure that emerges from immediate constituent analysis. However once formulated this way, the analysis becomes predictive because the rules of the grammar generate further sentences paired with hierarchical structure. Generative theory is thus good for capturing the productivity of human language(s). However, even with a meta-theory, we still need methods to choose between analyses and to choose the meta-theory. So we will focus primarily on linguistic analysis (and terminology) for now.

A language can be generated given its grammar $G = (NT, \Sigma, S, P)$, where NT is set of variables (non-

terminal symbols), Σ is set of terminal symbols, which appear at the end of generation, S is start symbol, and P is set of production rules. The corresponding language of G is $L(G)$.

Consider that various tuples are as given follows, where V' is variable symbols (also called non-terminal symbols),

$$NT = \{S, NP, N, VP, V, Art\}$$

$$\Sigma = \{boy, icecream, dog, bite, like, ate, the, a\},$$

$$\begin{aligned} P = \{&S \rightarrow NP \;\; VP, \\ &NP \rightarrow N \mid ART \;\; N, \\ &VP \rightarrow V \mid V \;\; NP, \\ &N \rightarrow boy \mid icecream \mid dog, \\ &V \rightarrow ate \mid like \mid bite, \\ &Art \rightarrow the \mid a\} \end{aligned}$$

Using above we can generate the following sentences:

The dog bites boy.

Boy bites the dog.

Boy ate icecream.

The dog bite the boy.

9.2.1 Structural Representation

It is convenient to represent the sentences as tree or a graph to help expose the structure of the constituent parts. For example, the sentence, 'the boy ate a icecream' can be represented as a tree shown in Fig. 9.1.

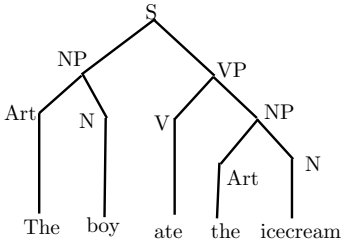


Figure 9.1: A syntax tree for "The boy ate icecream"

For the purpose of computation a tree must also be represented as a record, a list or some similar data structure. For example, the tree above is represented as a *list*:

```
(S (NP ((Art the)
        (N boy))
   (VP (V ate) (NP (Art the) (N Icecream)))))
```

A more extensive English grammar can be obtained with the addition of other constituencies such as prepositional phrases PP, adjectives ADJ, determiners DET, adverbs ADV, auxiliary verbs AUX, and many other features. Correspondingly, the other rewrite rules are followings.

$$
\begin{aligned}
&PP \rightarrow Prep \;\; NP, \\
&VP \rightarrow V \;\; ADV \mid V \;\; PP \mid V \;\; NP \;\; PP \mid AUX \;\; V \;\; NP \\
&Det \rightarrow Art \;\; ADJ \mid Art \\
&Art \rightarrow a \mid an \mid the
\end{aligned}
$$

These extensions allow the increase in complexity of the sentences, along with its expression power. For example, the following sentences.

The cruel man locked the dog in the house.

The laborious man worked to make some extra money.

9.3 Grammars and NL Parsing

To generate a sentence, the rules from production set P are applied sequentially starting from the beginning. However, we note that a grammar does not guarantee the generation of meaningful sentences, but generate only those that are structurally correct as per the production rules of the grammar. In fact, it is not always possible to formally characterize the natural languages with a simple grammar like above.

The grammars are defined by Chomsky hierarchy, as type $0, 1, 2, 3$ [noamcho]. The typical rewrite rules for type 1 grammar (called Context-Sensitive grammar) are:

$$
\begin{aligned}
S &\rightarrow aS \mid aAB \\
AB &\rightarrow BA \\
aA &\rightarrow ab \mid aa
\end{aligned}
\tag{9.1}
$$

In above, the uppercase letters are non-terminals and lowercase are terminals.

The type-2 grammars, also called Context-Free grammar, are:

$$
\begin{aligned}
&S \rightarrow aS \mid aSb \mid aB \mid aAB \\
&A \rightarrow a \\
&B \rightarrow b
\end{aligned}
\tag{9.2}
$$

The type 3 grammar (also called Regular grammar) is simplest having rewrite rules as:

$$S \rightarrow aS \mid a \tag{9.3}$$

Since, grammars of the types 1, 2, 3 are called context-sensitive, context-free, and regular grammars, respectively, and hence the corresponding names for languages also similar. The formal languages are mostly based on the type-2 languages. The type 0 and 1 are not much understood and difficult to implement.

Following are examples showing the production rules and corresponding parsed sentences:

$$\begin{aligned}
S &\rightarrow NP\ VP; && \text{I prefer a morning flight} \\
VP &\rightarrow V\ NP; && \text{Prefer a morning flight} \\
VP &\rightarrow V\ NP\ PP; && \text{Leaves Mumbai in the morning} \\
VP &\rightarrow V\ PP; && \text{Leaving on Tuesday} \\
PP &\rightarrow Prep\ NP; && \text{from New Delhi.}
\end{aligned}$$

The NP in the PP can be also be location, date, time or others.

The components of a sentence, i.e., noun, pronoun, adjective, adverb, determiner, preposition, and conjunction, are called parts-of-speech. Following are examples of Parts-of-Speech (POS).

$$\begin{aligned}
N &\rightarrow flights \mid breeze \mid trip \mid morning \mid ... \\
V &\rightarrow is \mid prefer \mid like \mid need \mid want \mid fly \\
Adj &\rightarrow cheapest \mid non\text{-}stop \mid first \mid latest \mid other \mid direct... \\
Pronoun &\rightarrow me \mid I \mid you \mid it \mid ... \\
Proper\text{-}N &\rightarrow Mumbai \mid Delhi \mid India \mid USA \mid ... \\
Det &\rightarrow a \mid an \mid the \mid this \mid these \mid those \mid ... \\
Prep &\rightarrow from \mid to \mid on \mid near \\
Conj &\rightarrow and \mid or \mid but
\end{aligned}$$

The following examples show the substitution rules along with values for each POS to be substituted, shown in parentheses.

$$\begin{aligned}
&NP \rightarrow Pronoun(I) \mid proper\text{-}N(Mumbai) \mid det\ Nomial(a\ flight) \mid N\ (flight) \\
&VP \rightarrow V(do) \mid V\ NP(\text{want a flight}) \mid V\ NP\ PP(\text{leaves Delhi in Morning}) \\
&PP \rightarrow Pre\ NP(\text{from Delhi})
\end{aligned}$$

Making use of above (substitution) rules, the figure 9.2 demonstrates the parsing of sentence "I prefer morning flight."

9.4 Transformational Grammars

The grammar discussed above produce different structures for different sentences, even though they have same meaning. For example,

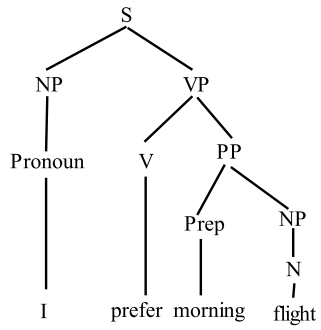


Figure 9.2: Parse-Tree for "I prefer morning flight"

Sentence-1: Ram gave Shyam a book.

Sentence-: A book was given by ram to Shyam.

In the sentence-2 above, the subject and object roles are switched. In the first, subject is Ram and object is Book, while in second sentence they are other way round. In sentence-2, it is an undesirable feature for machine processing of a language. In fact, sentences having same meaning should map to the same internal structures.

By adding some extra components, we can produce a single representation for sentences having the same meaning, through a series of transformations. This extended grammar is called *Transformational grammar*. In addition, the semantic and phonological components components, added as new, helps in interpreting the output of the syntactic components, as meaning and sound sequences. The transformations are tree manipulation rules, which are taken from dictionary, where words contain semantic featuring each of the lexicon.

Using transformational generative grammar, a sentence is analyzed in two stages:

1. Basic structure of the sentence is analyzed to determine the grammatical constitutional parts, which provides the structure of the sentence.

2. This is transformed into another form, where deeper semantic structure is determined.

The application of transformations is to produce a change from passive voice form of the sentence into active voice, change a question to declarative form, handle negations, and provide subject-verb agreement. The figure 9.3 shows: I. the active voice sentence, and III. passive voice converted into active voice. Note that, to arrive at III from II, we we exchange subject (a) and object (e) in II, auxiliary "was" is dropped, and verb "taught" is converted past to present tense, i.e., "teaches". Therefore, transformation from passive voice to active voice is nothing but, structural change in the parse-tree.

However, the transformational grammars are rarely used as computational models.

9.5 Sentence Level Constructions

The sentences can be classified as *declarative*, *imperative*, and *pragmatic*, as follows.

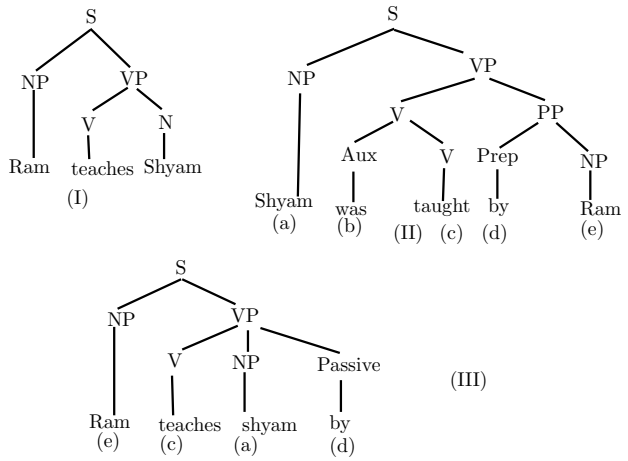


Figure 9.3: Transformation steps for passive format to active format

- **Declarative Sentences:** They have structure: `S` → `NP VP`.

- **Imperative Sentences:** These sentences begin with 'verb'. For example, the sentence "Show the lowest fare", and "List all the scores" are imperative sentences. The top level productions rules for generating these sentences are:

$$\texttt{S} \rightarrow \texttt{VP}$$
$$\texttt{VP} \rightarrow \texttt{V NP}$$

  The other substitutions rule for verb, we have discussed in above.

- **Pragmatic Sentences:** The examples of pragmatic sentences are:

  Do all these flights have stops?

  Can you give me the same information?

  What Airlines fly from Delhi?

  What flights do you have from Delhi to Mumbai?

The substitution rule for the pragmatic sentences is:

$$\texttt{S} \rightarrow \texttt{Aux NP VP} \tag{9.4}$$

Corresponding to the "What", the production rule is "`Wh-NP` → `What`". Hence, for the sentence, "What flights do you have from Delhi to Mumbai?", the first rule to be applied is "`S` → `Wh-NP Aux NP VP`". Many times, the longer sentences are conjuncted together using connectives, e.g., "I will fly to Delhi and Mumbai". The corresponding rule is "`S` → `NP` *and* `NP`". Similarly, there is "`S` → `S` *and* `D`", and "`VP` → `VP` *and* `VP`".

9.6 Parsing with Context-Free Grammars

The parse-trees are useful for:

1. Grammar checking of the sentence,
2. Parsing is an important intermediate stage in semantic analysis.
3. The parsing plays an important role in:
   (a) Mechanical translation,
   (b) Question answering
   (c) Information Extraction

9.6.1 Parsing is Search

A syntactic parser can be viewed as searching through the space of all possible parse-trees to find the correct parse-tree. Before we go through the steps of parsing, let us consider the following rules for grammar.

$$
\begin{aligned}
S &\rightarrow NP\ \ VP \\
S &\rightarrow Aux\ NP\ \ VP \\
S &\rightarrow VP \\
NP &\rightarrow Det\ \ Nom \\
Nom &\rightarrow Noun\ \ Noam \\
Nom &\rightarrow N \\
NP &\rightarrow \textit{proper-N} \\
VP &\rightarrow V \\
VP &\rightarrow V\ NP \\
Det &\rightarrow a \mid an \mid the \\
N &\rightarrow book \mid flight \mid meal \\
V &\rightarrow book \mid include \mid proper \\
Aux &\rightarrow Does \\
prep &\rightarrow from \mid to \mid on \\
\textit{Proper-N} &\rightarrow Mumbai \\
Nomial &\rightarrow Nomial\ \ PP
\end{aligned}
$$

The parse tree is shown in figure 9.4.

9.6.2 Top-Down parsing

The searching is carried out from the root node. The substitutions are carried out, and progressing sentence is compared with the input text sentence to determine whether the sentence generated progressively matches

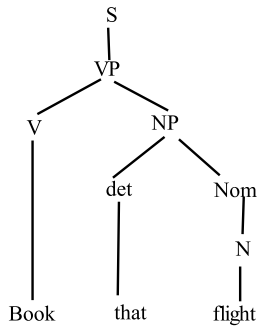


Figure 9.4: Parsing of: "Book that flight"

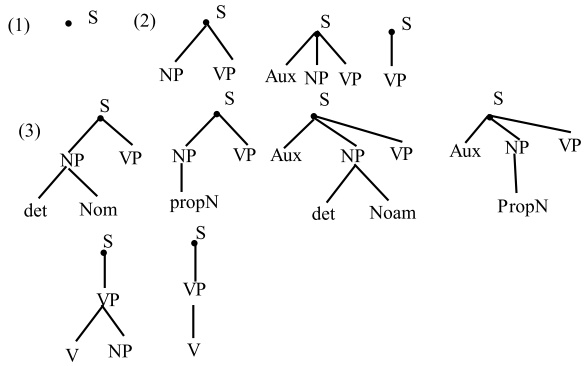


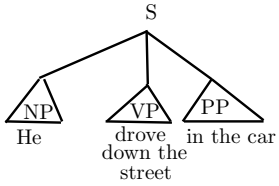
Figure 9.5: Top-down parsing of: "Book that flight"

with the original. The figure 9.5 demonstrates the steps for the top-down parsing for the sentence "Book that flight".

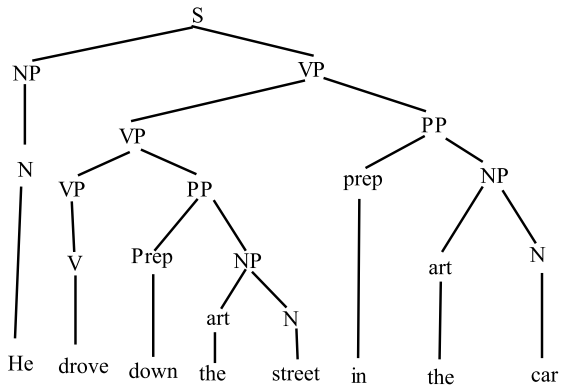
To carry out the top down parsing, we expand the tree at each level as shown in the figure. At each level, the trees whose leaves fails to match the input sentence, are rejected, leaving behind the trees that represent the successful parses. Going this way, ultimately get the sentence: "Book that flight."

9.7 Ambiguous Grammars

The ambiguous grammars have more than one parse-trees for the same sentence. Consider the sentence "He drove down the street in the Car". The two different parse-trees for this sentence are given in Fig. 9.6 and 9.7. A process for drawing the parse-trees is grouping the words to realize the structure in the sentence. Fig. 9.6a and 9.6b demonstrate grouping of the words and parsing for one parse-tree, while Fig. 9.7a and 9.7b demonstrate for other parse-tree. Note that the words "in" and "down" are both two prepositions, hence two separate PP subtrees in Fig. 9.7.

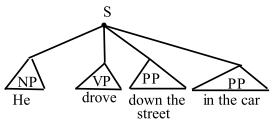


(a) Parsing-1: Grouping of the words for parsing

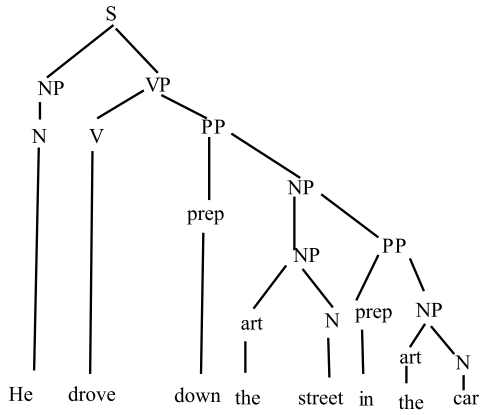


(b) Parsing-1 for : "He drove down the street in the car"

Figure 9.6: Parsing steps for : "He drove down the street in the car"



(a) Parsing-2: Grouping of the words for parsing



(b) Parsing-2 for : "He drove down the street in the car"

Figure 9.7: Parsing steps for : "He drove down the street in the car"

9.8 Probabilistic Parsing

During the discussion above, we drawn a sharp line between a correct and an incorrect parses, which was based whether a terminal node either matched or did not match the next word in the sentence. According to this parsing, a phrase is either acceptable or not. There are situations under which we can relax these requirements, e.g., when we cannot afford to try alternatives exhaustively. The examples are: analysis of connected speech, text segmentation and identification of words can never be done with complete certainty. At the best, one can say that certain sound is more probable than the other. Consequently, one may associate a fractional number with each terminal node, indicating the probability or quality of nodes below that, in respect of forming grammatically correct sentence [djjm02].

In the probabilistic parsing, the non-terminal nodes will be assigned some value, which are sophisticated enough to realize that syntactic and semantic restrictions are taken care of. Hence, a parser that aims to deliver the best analysis even if every analysis violates some constraint, must associate a measure of being grammatical (i.e., acceptable) with the analysis of portions of the sentence. The sentence ultimately associates a measure with the analyses of the complete sentence. As a matter of rule, it is possible generate

analysis of every sentence with a non-zero acceptability or matching probability, and then select the one having best analysis.

One simple parser of this type is "best-first" parser, which is based on modified form of standard *top-down serial* parsing algorithm for context-free grammars. The standard algorithm tries to generates one possible parse-tree until it gets stuck (i.e., on generation a terminal node which does not match the next word in the sentence). In case of stuck, it "backs up" to try another alternative. A *best-procedure*, is like a best-first search that tries all alternatives in parallel. A measure in numerical value is associated with each alternative path that indicates the likelihood, that this analysis matches the sentence processed so far, and it can be extended to the complete sentence analysis. At each progressive step, the path with the highest likelihood is extended. In the process, if the "measure" of current path falls below that of some other path, the parser shifts its attention to that of other path.

A CFG (context-free grammar) consists of,

terminal words $w^1, w^2, ..., w^V$,

non-terminals $N^1, N^2, ..., N^n$,

start symbol N^1, and

production rules $N^i \rightarrow \alpha^j$

where V is length of the sentence w, n number of non-terminals, α^j is sequence of terminals and non-terminals. Given this, we can define a generative PCFG (probabilistic context-free grammar) as,

terminal words $w^1, w^2, ..., w^V$,

non-terminals $N^1, N^2, ..., N^n$,

start symbol N^1, and

production rules $N^i \rightarrow \alpha^j$

where α^j is sequence of terminals and non-terminals. And can define the rule probabilities as,

$$\forall_i \sum_j P(N^i \rightarrow \alpha^j) = 1, \tag{9.5}$$

which shows that for each set of productions having same left side variable (N^i), the sum of probabilities is unity.

We consider that sentence is represented as sequence of words $w_1 w_2 ... w_m$, and $w_{ab} = w_a ... w_b$ is a subsequence. Let non-terminal N^i dominates sub-sequence $w_a ... w_b$ is represented by N^i_{ab}, .i.e., N_i is root of subtree having children sequence as $w_a ... w_b$. Let $N^i \Rightarrow^* \alpha$. We represent the probability of sentence w_{1n} as,

$$P(w_{1n}) = \sum_t P(w_{1n,t}) \tag{9.6}$$

where t is parse tree of sentence w_{1n}.

Example. Construct a correct parse-tree for the sentence "I saw an astronomer with telescope", for the following PCFG:

Rule	Probability	Rule	Probability
$S \to NP\ \ VP$	1.0	$NP \to NP\ \ PP$	0.20
$PP \to Prep\ \ NP$	1.0	$NP \to N$	0.45
$VP \to V\ \ NP$	0.7	$NP \to Art\ \ N$	0.35
$VP \to VP\ \ PP$	0.3	$N \to telescope$	0.25
$Prep \to with$	1.0	$N \to astronomer$	0.15
$Art \to an$	1.0	$N \to I$	0.60
$V \to saw$	1.0		

The sentence above can be generated using two different parse-trees, say t_1 (see Fig. 9.8) and t_2 (see Fig. 9.9).

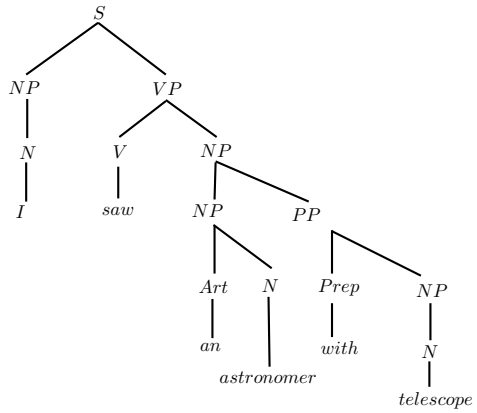


Figure 9.8: Probabilistic parse-tree t_1 for the sentence "I saw an astronomer with telescope"

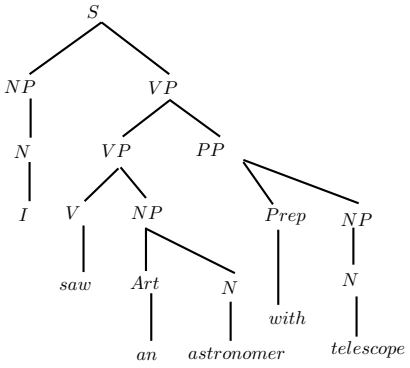


Figure 9.9: Probabilistic parse-tree t_2 for the sentence "I saw an astronomer with telescope"

In this problem, the symbols are as follows:

- Terminals: I, saw, an, astronomer, with, telescope.
- Non-terminals: *S, NP, VP, PP, V, Prep, Art, N*

- Start Symbol: S

Probability for parse tree t_1 is product of all the probabilities of rules used, which starting from top, and going down through all the levels is given by:

$$\begin{aligned} P(t_1) =& 1.0 \times 0.45 \times 0.7 \times 0.60 \times 1.0 \times 0.20 \times 0.35 \times 1.0 \times 1.0 \times 0.45 \times 0.25 \\ =& 0.001488375 \end{aligned}$$

Similarly, the probability for parse tree t_2 is given by:

$$\begin{aligned} P(t_2) =& 1.0 \times 0.45 \times 0.3 \times 0.60 \times 1.0 \times 0.35 \times 1.0 \times 0.45 \times 1.0 \times 0.15 \times 0.25 \\ =& 0.00047840625 \end{aligned}$$

Naturally, the parse tree t_1 is correct, as probability of its construction is higher. The tree t_1 indicates that the observer (I) is seeing an astronomer carrying telescope, while t_2 has semantics which indicates that observer (I) is seeing an astronomer with the help of telescope, which obviously is incorrect[1].

However, the accuracy of probability of each tree depends on the accuracy of the probabilities of rules used. The probabilities associated with the rules are made available from the the statistics of semantics from large corpus of natural language text. □

The probabilistic parsing provides a solution to ambiguity in language and grammar, as it is conclusive from above example. But, this is not necessarily complete, but in partial. The other benefit is that it gives a *probabilistic language model*.

[1] One would see the stars through telescope and not the astronomer through a telescope !