CHAPTER 2

LANGUAGE MODELLING

CHAPTER OVERVIEW

The domain of language is quite vast. It presents an almost infinite number of sentences to the reader (or computer). To handle such a large number of sentences, we have to create a model of the domain, which can subsequently be simplified and handled computationally. A number of language models have been proposed. We introduce some of these models in this chapter. To create a general model of any language is a difficult task. There are two approaches for language modelling. One is to define a grammar that can handle the language. The other is to capture the patterns in a grammar language statistically. This chapter has a mixed approach. It gives a glimpse of both grammar-based model and statistical language model. These include lexical functional grammar, government and binding, Paninian grammar, and *n*-gram based model.

2.1 INTRODUCTION

Why and how do we model a language? This question has been discussed by linguists since 500 BC. Computational linguists also have to confront this question. It is obvious that our purpose is to understand and generate natural languages from a computational viewpoint. One approach can be to just take a language, try to understand every word and sentence of it, and then come to a conclusion. This approach has not succeeded as there are difficulties at each stage, which we will understand as we go through this book. An alternative approach is to study the grammar of various languages, compare them, and if possible, arrive at reasonable models that facilitate our understanding of the problem and designing of natural-language tools.

A model is a description of some complex entity or process. A language model is thus a description of language. Indeed, natural language is a complex entity and in order to process it through a computer-based program, we need to build a representation (model) of it. This is known

as *language modelling*. Language modelling can be viewed either as a problem of grammar inference or a problem of probability estimation. A grammar-based language model attempts to distinguish a grammatical sentence from a non-grammatical (ill-formed) one, whereas a probabilistic language model attempts to identify a sentence based on a probability measure, usually a maximum likelihood estimate. These two viewpoints have led to the following categorization of language modelling approaches.

Grammar-based language model

A grammar-based approach uses the grammar of a language to create its model. It attempts to represent the syntactic structure of language. Grammar consists of hand-coded rules defining the structure and ordering of various constituents appearing in a linguistic unit (phrase, sentence, etc.). For example, a sentence usually consists of noun phrase and a verb phrase. The grammar-based approach attempts to utilize this structure and also the relationships between these structures.

Statistical language modelling

The statistical approach creates a language model by training it from a corpus. In order to capture regularities of a language, the training corpus needs to be sufficiently large. Rosenfeld (1994) pointed out:

> *Statistical language modelling (SLM) is the attempt to capture regularities of natural language for the purpose of improving the performance of various natural language applications.*

Statistical language modelling is one of the fundamental tasks in many NLP applications, including speech recognition, spelling correction, handwriting recognition, and machine translation. It has now found applications in information retrieval, text summarization, and question answering also. A number of statistical language models have been proposed in literature. The most popular of these are the *n*-gram models. We discuss this model in Section 2.3. The following section discusses various grammar-based models.

2.2 VARIOUS GRAMMAR-BASED LANGUAGE MODELS

Various computational grammars have been proposed and studied, e.g., transformational grammar (Chomsky 1957), lexical functional grammar (Kaplan and Bresnan 1982), government and binding (Chomsky 1981), generalized phrase structure grammar (Gazdar et al. 1985), dependency grammar, paninian grammar, and tree-adjoining grammar (Joshi 1985).

This section focuses on lexical functional grammar (LFG), generalized phrase structure grammar (GPSG), government and binding (GB), and Paninian grammar (PG) and introduces various approaches to understand a language in a grammatical and rule-based format. It also introduces the dominant approaches to create statistical models of language and grammar.

2.2.1 Generative Grammars

In 1957, in his book on *Syntactic Structures*, Noam Chomsky wrote that we can generate sentences in a language if we know a collection of words and rules in that language. Only those sentences that can be generated as per the rules are grammatical. This point of view has dominated computational linguistics and is appropriately termed generative grammar. The same idea can be used to model a language. If we have a complete set of rules that can generate all possible sentences in a language, those rules provide a model of that language. Of course, we are talking only about the syntactical structure of language here.

Language is a relation between the sound (or the written text) and its meaning. Thus, any model of a language should also deal with the meaning of its sentences. As seen earlier, we can have a perfectly grammatical but meaningless sentence.

In this chapter, we will assume that grammars are a type of language models.

2.2.2 Hierarchical Grammar

Chomsky (1956) described classes of grammars in a hierarchical manner, where the top layer contained the grammars represented by its sub classes. Hence, Type 0 (or unrestricted) grammar contains Type 1 (or context-sensitive grammar), which in turn contains Type 2 (context-free grammar) and that again contains Type 3 grammar (regular grammar). Although this relationship has been given for classes of formal grammars, it can be extended to describe grammars at various levels, such as in a class-sub class (embedded) relationship.

2.2.3 Government and Binding (GB)

As discussed in Chapter 1, a common viewpoint taken by linguists (not computational linguists, however) is that the structure of a language (or how well its sentences are formed) can be understood at the level of its meaning, particularly while resolving structural ambiguity. However, the sentences are given at the syntactical level and the transformation from meaning to syntax or vice versa is not well understood.

Transformational grammars assume two levels of existence of sentences—one at the surface level and the other at the deep root level (this should not be confused with the meaning level). Government and binding (GB) theories have renamed them as s-level and d-level, and identified two more levels of representation (parallel to each other) called *phonetic form* and *logical form*. According to GB theories, language can be considered for analysis at the levels shown in Figure 2.1.

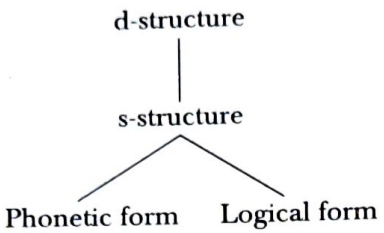


Figure 2.1 Different levels of representation in GB

If we describe language as the representation of some 'meaning' in a 'sound' form, then according to Figure 2.1, these two ends are the logical form (LF) and phonetic form (PF) respectively. The GB is concerned with LF, rather than PF. Chomsky was the first to put forward a GB theory (Peter Sells 1985).

Transformational grammars have hundreds of rewriting rules, which are generally language-specific and also construct-specific (say, different rules for assertive and interrogative sentences in English, or for active and passive voice sentences). Generation of a complete set of coherent rules may not be possible. The GB envisages that if we define rules for structural units at the deep level, it will be possible to generate any language with fewer rules. These deep-level structures are abstractions of noun-phrase, verb-phrase, etc., and common to all languages. It is possible to do if, as GB theory states, a child learns its mother tongue because the human mind is 'hard-wired' with some universal grammar rules. Thus, the data enters the mind and its abstract structure gives rise to actual phonetic structures. The existence of deep level, language-independent, abstract structures, and the expression of these in surface level, language-specific structures with the help of simple rules is the main concern of GB theories. Let us take an example to explain d- and s-structures.

Example 2.1

Mukesh was killed. (2.1)

(i) In transformational grammar, this can be represented as S-NP Aux VP as given below:

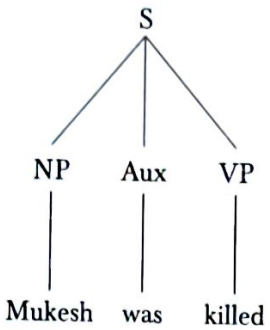
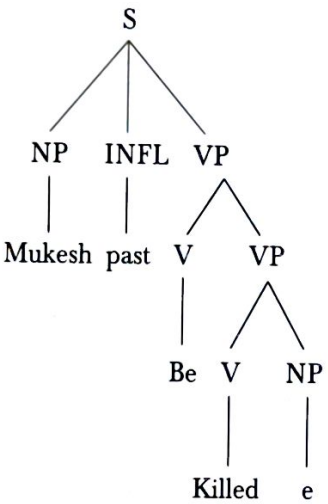


Figure 2.2 TG representation of sentence (2.1)

(ii) In GB, the s-structure and d-structure are as follows:



Mukesh was killed
(e) killed Mukesh
(e) past kill Mukesh

INFL: Inflection
NP: Noun phrase
VP: Verb phrase
e: empty

Figure 2.3 Surface structure of sentence (2.1) in GB

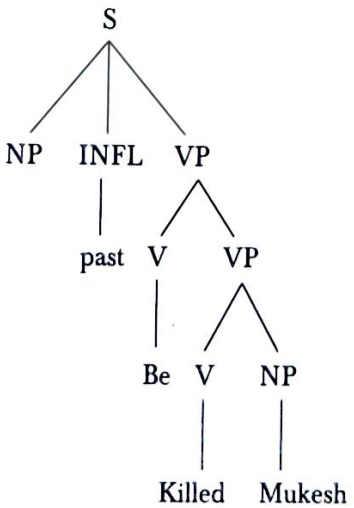


Figure 2.4 Deep structure of sentence (2.1) in GB

Components of GB

Government and binding (GB) comprises a set of theories that map the structures from d-structure to s-structure and to logical form (LF) (leaving aside the phonetic form). A general transformational rule called 'Move α' is applied at d-structure level as well as at s-structure level. This can move constituents at any place if it does not violate the constraints put by several theories and principles. Hence, in its simplest from GB can be represented by Figure 2.5.

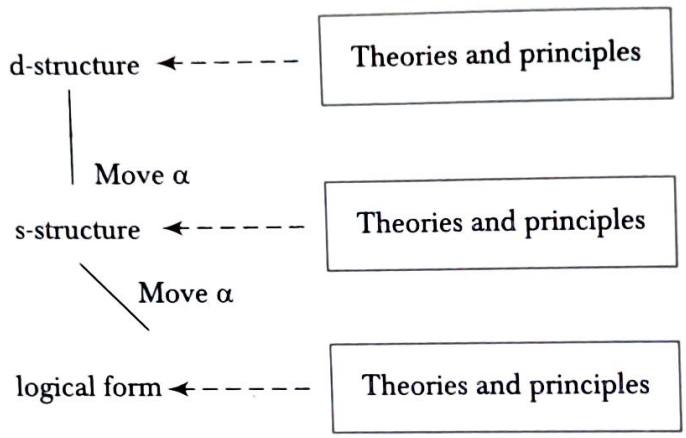


Figure 2.5 Components of GB

Hence, GB consists of 'a series of modules that contain constraints and principles' (Sells 1985) applied at various levels of its representations and the transformation rule, Move α. Before elaborating on these modules—which include X-bar theory, projection principle, θ-theory and θ-criterion, C-command and government, case theory, empty category principle (ECP), and binding theory—we discuss the general characteristics of GB.

The GB considers all three levels of representations (d-, s-, and LF) as syntactic, and LF is also related to meaning or semantic-interpretive mechanisms. However, GB applies the same Move α transformation to map d-levels to s-levels or s-levels to LF level. LF level helps in quantifier scoping and also in handling various sentence constructions such as passive or interrogative constructions. An example of LF representation may be helpful.

Example 2.2 Consider the sentence:

Two countries are visited by most travellers. (2.2)

Its two possible logical forms are:

LF1: $[_S$ Two countries are visited by $[_{NP}$ most travellers$]]$
LF2: Applying Move α
$[_{NP}$ Most travellers$_i$ $]$ $[_S$ two countries are visited by $e_i]$

In LF1, the interpretation is that most travellers visit the same two countries (say, India and China). In LF2, when we move [most travellers] outside the scope of the sentence, the interpretation can be that most travellers visit two countries, which may be different for different travellers.

One of the important concepts in GB is that of constraints. It is the part of the grammar which prohibits certain combinations and movements; otherwise Move α can move anything to any possible position. Thus, GB is basically the formulation of theories or principles which create constraints to disallow the construction of ill-formed sentences. To account for cross-lingual constraints of similar type, GB can specify that 'a constituent cannot be moved from position X' (where X can have value X_1 in one language, X_2 in another, and so on). These rules are so general and language-independent that 'language-particular details of description typically go uncharted in GB' (Sells 1985).

Figure 2.5 showed the application of various theories and principles at three different levels of representations in GB. Figure 2.6 mentions these explicitly to understand the organization of GB.

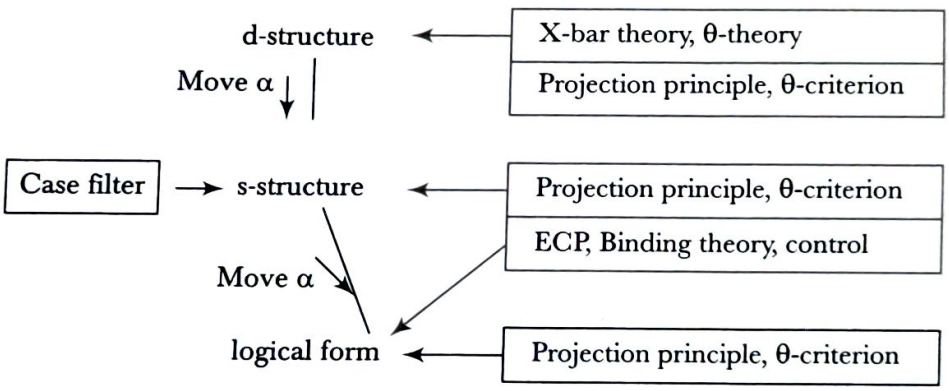


Figure 2.6 Organization of GB (adapted from Peter Sells 1985)

$\bar{X}$ *Theory*

The $\bar{X}$ theory (pronounced X-bar theory) is one of the central concepts in GB. Instead of defining several phrase structures and the sentence structure with separate sets of rules, $\bar{X}$ theory defines them both as maximal projections of some head. In this manner, the entities defined become language independent. Thus, noun phrase (NP), verb phrase (VP), adjective phrase (AP), and prepositional phrase (PP) are maximal projections of noun (N), verb (V), adjective (A), and preposition (P) respectively, and can be represented as head X of their corresponding phrases (where X = {N, V, A, P}). Not only that, even the sentence

structure (S′, which is projection of sentence) can be regarded as the maximal projection of inflection (INFL). The GB envisages projections at two levels—first the projection of head at semi-phrasal level, denoted by $\bar{X}$, and then the second maximal projection at the phrasal level, denoted by $\bar{\bar{X}}$.

For sentences, the first level projection is denoted as S and the second level maximal projection is denoted by S′. We now illustrate phrase and sentence representations with the help of examples.

Example 2.3 Figure 2.7 depicts the general and particular structures with examples. We see the general structure in Figure 2.7(a).

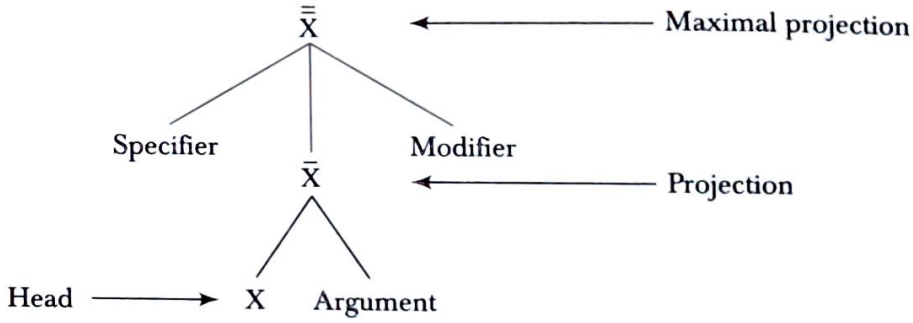


Figure 2.7(a) General phrase and sentential structure

Next, we consider the representation of the NP, *the food in a dhaba*. This is followed by the representation of VP, AP, and PP structure in Figure 2.7(c–e); and finally Figure 2.7(f) shows the representation of a sentence.

1. NP: the food in a dhaba

   $[_{NP}\ the\,[_{N}\ food\,]_{PP}\,[in\ a\ dhabha]]$

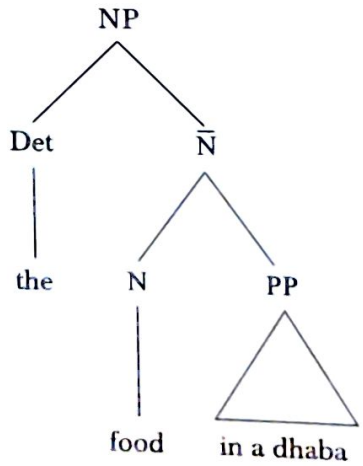


Figure 2.7(b) NP structure

2. VP: ate the food in a dhaba

$$\left[_{VP}\left[_{\bar{V}}[_{V}\ ate][_{NP}\ the\ food]\right][_{PP}\ in\ a\ dhaba]\right]$$

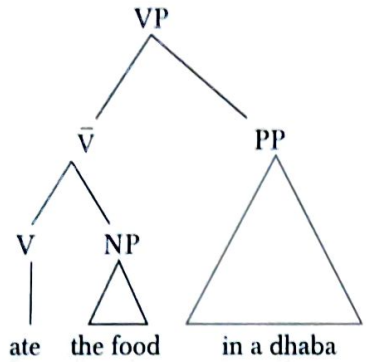


Figure 2.7(c) VP structure

3. AP: very proud of his country

$$\left[_{AP}[_{Deg}\ very][_{\bar{A}}[_{A}\ proud][_{PP}\ of\ his\ country]]\right]$$

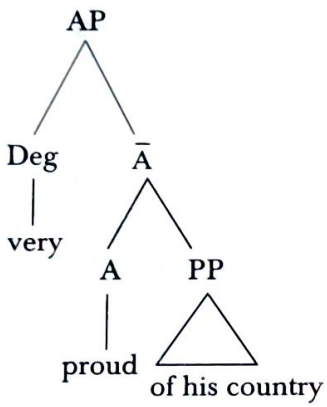


Figure 2.7(d) AP structure

4. PP: in a dhaba

$$\left[_{PP}\left[_{\bar{P}}[_{P}\ in]\left[_{NP}[_{\mathrm{Det}}\ a][_{N}\ dhabha]\right]\right]\right]$$

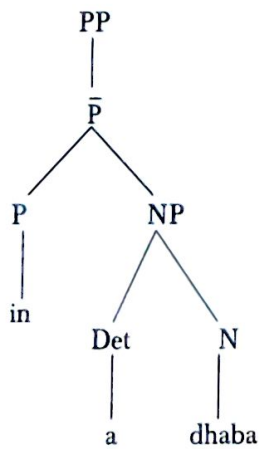


Figure 2.7(e) PP structure

5. S: that she ate the food in a dhaba

$[_{\bar{S}}[_{COMP}\ that][_{S}[_{Det}\ she][_{INFL}\ past][_{VP}\ ate\ the\ food\ in\ a\ dhaba]]]$

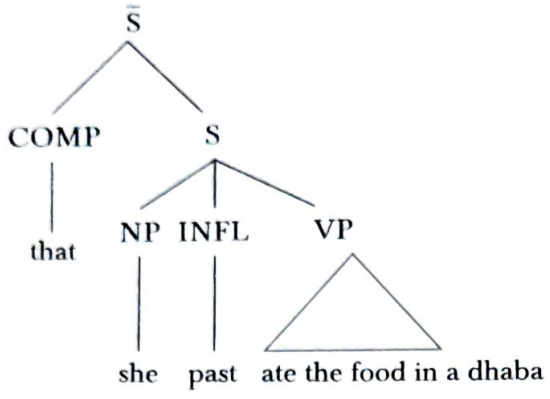


Figure 2.7(f) Maximal projection of sentence structure

As shown in Figure 2.7(f), the sentence is considered to be the head of INFL and the projection of sentence is denoted by $\bar{S}$, which has the specifier as complementizer (COMP).

Sub-categorization It is to be noted that GB does not consider traditional phrase structure as an appropriate device for defining language constructs. It places the burden of ascertaining well-formedness to sub-categorization frames of heads. In principle, any maximal projection can be the argument of a head, but sub-categorization is used as a filter to permit various heads to select a certain subset of the range of maximal projections. For example, we know that the verb 'eat' can sub-categorize for NP, whereas the verb 'sleep' cannot. Hence, 'ate food' is well-formed, but the sentence '*slept the bed' is not. GB claims that defining phrase structures as head projections and sub-categorization helps ensure well-formed structures, even at the sentence level.

As 'verb' is not the head of the sentence, it cannot sub-categorize for 'subject NP', while it can perfectly sub-categorize for 'object NP'. This explains, to certain extent, the 'subject/object' asymmetry (Sells 1985).

Projection Principle

The projection principle, a basic notion in GB, places a constraint on the three syntactic representations and their mapping from one to the other. The principle states that representations at all syntactic levels (i.e., d-level, s-level, and LF level) are projections from the lexicon. Thus, lexical

*will precede the incorrect sentences.

properties of categorical structure (sub-categorization) must be observed at each level.

This can be understood with an example. Suppose 'the object' is not present at d-level, then another NP cannot take this position at s-level. This principle, in conjunction with the possibility of presence of empty category and other theories (like Binding Theory) ensures correct movement and well-formed structure.

Theta Theory (θ-Theory) or The Theory of Thematic Relations

As discussed earlier, 'sub-categorization' only places a restriction on syntactic categories which a head can accept. GB puts another restriction on the lexical heads through which it assigns certain roles to its arguments. These roles are pre-assigned and cannot be violated at any syntactical level as per the projection principle. These role assignments are called theta-roles and are related to 'semantic-selection'.

Theta-role and Theta-criterion There are certain thematic roles from which a head can select. These are called θ-roles and they are mentioned in the lexicon, say for example the verb 'eat' can take arguments with θ-roles '(Agent, Theme)'. Agent is a special type of role which can be assigned by a head to outside arguments (external arguments) whereas other roles are assigned within its domain (internal arguments). Hence in 'Mukesh ate food', the verb 'eat' assigns the 'Agent' role to 'Mukesh' (outside VP) and 'Theme' (or 'patient') role to 'food'. As roles are assigned based on the syntactic positions of the arguments, it is important that there should be a match between the number of roles and number of arguments as depicted by θ-criterion.

Theta-Criterion states that 'each argument bears one and only one θ-role, and each θ-role is assigned to one and only one argument' (Sells 1985). Thus, each argument will have a unique θ-role and cannot be moved to a position where it may acquire another θ-role.

In GB, d-structure is conceived as some kind of 'pure' representation of arguments and hence, θ-roles are assigned at d-level only, whereas theta-criterion is applied at all the three levels, as shown in Figure 2.6.

C-command and Government

As 'Government' is a special case of 'C-command', we will first define C-command.

C-command C-command defines the scope of maximal projection. It is a basic mechanism through which many constraints are defined on Move

α. If any word or phrase (say α or β) falls within the scope of and is determined by a maximal projection, we say that it is dominated by the maximal projection. Now, if there are two structures α and β related in such a way that 'every maximal projection dominating α dominates β', we say that α C-commands β, and this is the necessary and sufficient condition (iff) for C-command.

The definition of C-command does not include all maximal projections dominating β, only those dominating α. If we put this extra constraint, it becomes a kind of mutual C-command (Sells 1985), called government.

Government

α governs β iff:
α C-commands β

α is an X (head, e.g., noun, verb, preposition, adjective, and inflection), and every maximal projection dominating β dominates α.

Thus no maximal projection can intervene between the governor and governee. In GB literature, this has been stated as: 'Maximal projections are barriers to government.'

Movement, Empty Category, and Co-indexing

Briefly let us discuss Move α. In GB, Move α is described as 'move anything anywhere', though it provides restrictions for valid movements.

In GB, the active to passive transformation is the result of NP movement as shown in sentence (2.3). Another well-known movement is the wh-movement, where wh-phrase is moved as follows.

What did Mukesh eat ? (2.3)
[Mukesh INFL eat what]

As discussed in the projection principle, lexical categories must exist at all the three levels. This principle, when applied to some cases of movement leads to the existence of an abstract entity called empty category. In GB, there are four types of empty categories, two being empty NP positions called wh-trace and NP trace, and the remaining two being pronouns called small 'pro' and big 'PRO'. This division is based on two properties—anaphoric (+a or –a) and pronominal (+p or –p).

Wh-trace	–a, –p
NP-trace	+a, –p
small 'pro'	–a, +p
big 'PRO'	+a, +p

Co-indexing is the indexing of the subject NP and AGR (agreement) at d-structure which are preserved by Move α operations at s-structure.

When an NP-movement takes place, a trace of the movement is created by having an indexed empty category (e_i) from the position at which the movement began to the corresponding indexed NP, i.e. NP_i. All A-positions (argument positions) at s-level are also freely indexed. These categories and indices are used to define Binding Theory.

It is interesting to note that for defining constraints to movement, the theory identifies two positions in a sentence. Positions assigned θ-roles are called θ-positions, while others are called $\bar{\theta}$-positions.

In a similar way, core grammatical positions (where subject, object, indirect object, etc., are positioned) are called A-positions (arguments positions), and the rest are called $\bar{A}$-positions.

Binding Theory

Binding is defined by Sells (1985) as follows:

α binds β iff
α C-commands β, and
α and β are co-indexed

As we noticed in sentence (2.1),

[e_i INFL kill Mukesh]
[$Mukesh_i$ was killed (by e_i)]
Mukesh was killed.

Empty clause (e_i) and Mukesh (NP_i) are bound. This theory gives a relationship between NPs (including pronouns and reflexive pronouns).

Now, binding theory can be given as follows:

(a) An anaphor (+a) is bound in its governing category.
(b) A pronominal (+p) is free in its governing category.
(c) An R-expression (–a, –p) is free.

This theory applies to binding at A-positions. Governing category is the local domain (the smallest only) NP or S containing it (G or p or R-expression) and its governor.

Example 2.4

A: $Mukesh_i$ knows $himself_i$
B: $Mukesh_i$ believes that Amrita knows him_i
C: Mukesh believes that $Amrita_j$ knows $Nupur_k$

Similar rules apply on empty categories also:

NP-trace: +a, –p: $Mukesh_i$ was killed e_i
wh-trace: –a, –p: Who_i does he_i like e_i

Empty Category Principle (ECP)

We have already defined 'government'. Now, let us define 'proper government':

α properly governs β iff:
α governs β and α is lexical (i.e. N, V, A, or P) or
α locally $\bar{A}$-binds β

The ECP says 'A trace must be properly governed'.

This principle justifies the creation of empty categories during NP trace and wh-trace and also explains the subject/object asymmetries to some extent. As in the following sentences:

(a) $What_i$ do you think that Mukesh ate e_i?
(b) $What_i$ do you think Mukesh ate e_i?

Bounding and Control Theory

There are many other types of constraints on Move α. It is not possible to explain all of them here, for details, see Peter Sells (1985).

In English, the long distance movement for complement clause can be explained by bounding theory if NP and S are taken to be bounding nodes. The theory says that the application of Move α may not cross more than one bounding node. The theory of control involves syntax, semantics, and pragmatics. As stated previously, the empty category 'PRO' (+a, +p) behaves as an anaphor sometimes, when it is the subject of the clausal complement to verbs such as decide and try. However, it behaves as pronoun with some other verbs.

Case Theory and Case Filter

In GB, case theory deals with the distribution of NPs and mentions that each NP (with the possible exception of a few empty categories) must be assigned a case. In English, we have the nominative, objective, genitive, etc., cases, which are assigned to NPs at particular positions. Indian languages are rich in case-markers, which are carried even during movements.

Case Filter An NP is ungrammatical if it has phonetic content or if it is an argument (with the exception of big 'PRO') and is not case-marked. Phonetic content here, refers to some physical realization, as opposed to empty categories. Thus, case filters restrict the movement of NP at a position which has no case assignment. It works in a manner similar to that of the θ-criterion.

In short, GB presents a model of the language which has three levels of syntactic representation. It assumes phrase structures to be the maximal

projection of some lexical head and in a similar fashion, explains the structure of a sentence or a clause. It assigns various types of roles to these structures and allows them a broad kind of movement called Move α. It then defines various types of constraints which restrict certain movements and justifies others. GB gives a new insight for the modelling of languages, although the Chomskian Minimalist Programme has superseded GB.

2.2.4 Lexical Functional Grammar (LFG) Model

This section presents those features of LFG that throw a light on language modelling. For the details of lexical functional grammar, readers are encouraged to seek Darlymple et al. (1995).

Unlike GB, LFG represents sentences at two syntactic levels—constituent structure (c-structure) and functional structure (f-structure). Based on Woods' *Augmented Transition Networks* 1970, which used phrase structure trees to represent the surface structure of sentences and the underlying predicate–argument structure, Kaplan (1975a, b) proposed a concrete form for the register names and values (used in ATN implementation), which became the functional structures in LFG. On the other hand, Bresnan (1976a, 1977) was more concerned with the problem of explaining some linguistic issues, such as active/passive and dative alternations, in transformational approach. She proposed that such issues can be dealt with by using lexical redundancy rules. The unification of these two diverse approaches (with a common concern) led to the development of the LFG theory, which was presented as *Lexical Functional Grammar: A Formal System for Grammatical Representation* in 1982.

The LFG is a formalism that is both computationally and linguistically motivated and provides precise algorithms for linguistic issues it can handle. The term 'lexical functional' is composed of two terms: the 'functional' part is derived from 'grammatical functions', such as subject and object, or roles played by various arguments in a sentence. The 'lexical' part is derived from the fact that the lexical rules can be formulated to help define the given structure of a sentence and some of the long distance dependencies, which is difficult in transformational grammars.

C-structure and f-structure in LFG

As LFG is aimed at providing exact computational algorithms, it provides well-defined objects called constituent structure (c-structure) and functional structure (f-structure). The c-structure is derived from the usual phrase and sentence structure syntax, as in CFG (discussed in Chapter 4). However,

as the grammatical-functional role cannot be derived directly from phrase and sentence structure, functional specifications are annotated on the nodes of c-structure, which when applied on sentences, results in f-structure. Hence, f-structure is the final product which encodes the information obtained from phrase and sentence structure rules and functional specifications.

Let us consider an example.

Example 2.5

She saw stars in the sky.

CFG rules to handle this sentence are:

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ \{NP\}\ \{NP\}\ PP^{*}\ \{S'\}$$
$$PP \rightarrow P\ NP$$
$$NP \rightarrow Det\ N\ \{PP\}$$
$$S' \rightarrow Comp\ S$$

where

S:	sentence	V:	verb
P:	preposition	N:	noun
S′:	clause	Comp:	complement

{ } optional

*: Phrase can appear any number of times including blank.

When annotated with functional specifications, the rules become:

Rule 1: $S \rightarrow \underset{\uparrow \text{subj} = \downarrow}{NP}\ \underset{\uparrow = \downarrow}{VP}$

Rule 2: $VP \rightarrow V\ \underset{\uparrow \text{obj} = \downarrow}{\{NP\}}\ \underset{\uparrow \text{obj 2} = \downarrow}{\{NP\}}\ \underset{\uparrow (\downarrow \text{case}) = \downarrow}{PP^{*}}\ \underset{\uparrow \text{comp} = \downarrow}{\{S'\}}$

Rule 3: $PP \rightarrow P\ \underset{\uparrow \text{obj} = \downarrow}{NP}$

Rule 4: $NP \rightarrow \{\text{Det}\}\ N\ \underset{\uparrow \text{Adjunct} = \downarrow}{\{PP\}}$

Rule 5: $S' \rightarrow \text{Comp}\ \underset{\uparrow = \downarrow}{S}$

Here, ↑ (up arrow) refers to the f-structure of the mother node that is on the left hand side of the rule. The ↓ (down arrow) symbol refers to the f-structure of the node under which it is denoted.

Hence, in Rule 1, (↑ subj = ↓) indicates that the f-structure of the first NP goes to the f-structure of the subject of the sentence, while (↑ = ↓) indicates that the f-structure of the VP node goes directly to the f-structure

of the sentence VP. Similarly, in Rule 2, the f-structure of VP is defined by the lexical item V, the two optional NPs, any number of PPs, and the optional clause(S′). The f-sructure of V can be obtained from the lexicon itself. All terminals in LFG can be thought of as annotated with ↑ = ↓. The NPs can function as object and object 2 of the sentence, and their f-structures are obained using f-sructure of Obj and Obj_2. '↑ (↓case) = ↓' in rule 2 indicaes that the f-sructure of the PP and the case of PP (some literature refers it as P case) determines the f-structure of VP. 'Comp' refers to the compliment in a sentence, e.g., 'He said *that* she is powerful'.

Let us see first the lexical entries of various words in the sentence.

She saw stars. (2.4)

She N (↑ Pred) = 'PRO'
(↑ Pers) = 3
(↑ Num) = SG
(↑ Gen) = FEM
(↑ Case) = NOM

Saw V ↑ Pred = 'see < (↑ Subj) (↑ Obj) >'
(↑ Tense = PAST)

Stars N ↑ Pred = 'Star'
↑ Pers = 3
↑ Num = PL

This will lead to the c-structure shown in Figure 2.8.

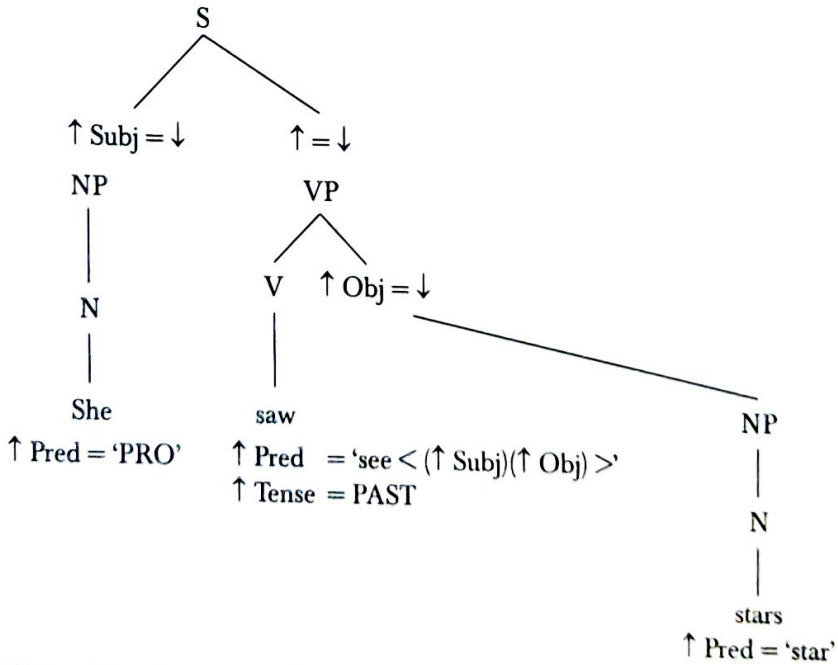


Figure 2.8 C-structure of sentence (2.4)

Finally, the f-structure is the set of attribute–value pairs, represented as

$$\begin{bmatrix} \text{subj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{SG} \\ \text{Gen} & \text{FEM} \\ \text{Case} & \text{NOM} \\ \text{Pred} & \text{'PRO'} \end{bmatrix} \\ \text{obj} & \begin{bmatrix} \text{Pers} & 3 \\ \text{Num} & \text{PL} \\ \text{Pred} & \text{'Star'} \end{bmatrix} \\ \text{Pred} & \text{'see'} <(\uparrow \text{subj})\ (\uparrow \text{obj})> \end{bmatrix}$$

It is interesting to note that the final f-structure is obtained through the unification of various f-structures for subject, object, verb, complement, etc. This unification is based on the functional specifications of the verb, which predicts the overall sentence structure.

LFG requires that all possible structures corresponding to passive constructs, dative constructs, etc., must be specified. If the given sentence does not match the specifications, it is said to be ill-formed. LFG imposes three conditions on f-structure (Sells 1985).

Consistency In a given f-structure, a particular attribute may have at the most one value. Hence, while unifying two f-structures, if the attribute Num has value SG in one and PL in the other, it will be rejected.

Completeness A function is called governable if it appears within the Pred value of some lexical form, e.g., Subj, Obj, and Obj 2. Adjunct is not a governable function.

When an f-structure and all its subsidiary f-structures (as the value of any attribute of f-structure can again contain other f-structures) contain all the functions that their predicates govern, then and only then is the f-structure complete. For example, since the predicate 'see < (↑ Subj) (↑ Obj) >' contains an object as its governable function, a sentence like 'He saw' will be incomplete.

Coherence Coherence maps the completeness property in the reverse direction. It requires that all governable functions of an f-structure, and all its subsidiary f-structures, must be governed by their respective predicates. Hence, in the f-structure of a sentence, an object cannot be taken if its verb does not allow that object. Thus, it will reject the sentence, 'I laughed a book.'

The completeness and coherence conditions are counterparts of θ-criterion in GB theory.

Lexical Rules in LFG

Different theories have different kinds of lexical rules and constraints for handling various sentence-constructs (active, passive, dative, causative, etc.). In GB, to express a sentence in its passive form, the verb is changed to its participial form and the ability of the verb to assign case and external (Agent) θ-role is taken away. In LFG, the verb is converted to the participial form, but the sub-categorization is changed directly. Consider the following example:

Active: Tara ate the food.
Passive: The food was eaten by Tara.
Active: ↑ Pred = 'eat < (↑ Subj) (↑ Obj) >'
Passive: ↑ Pred = 'eat < (↑ Obl_{ag}) (↑ Subj) >'

Here, Obl_{ag} represents oblique agent phrase. Similar rules can be applied in active and dative constructs for the verbs that accept two objects.

Active: Tara gave a pen to Monika.
Passive: Tara gave Monika a pen.
Active: ↑ Pred = 'give < (↑ Subj) (↑ Obj_2) (↑ Obj) >'
Passive: ↑ Pred = 'give < (↑ Subj) (↑ Obj) (↑ Obl_{go})>'

Here, Obl_{go} stands for oblique goal phrase. Similar rules are also applicable to the process of causativization. This can be seen in Hindi, where the verb form is changed as follows:

हँसना	$\xrightarrow{\text{causativization}}$	हँसाना
Laugh		Laugh-cause-past
		made to laugh

Example 2.6

Active: तारा हँसी
Taaraa hansii
Tara laughed

Causative: मोनिका ने तारा को हँसाया
Monika ne Tara ko hansaayaa
Monika Subj Tara Obj laugh-cause-past
Monika made Tara to laugh.

Active: ↑ Pred = 'Laugh <↑ Subj>'
Causative: ↑ Pred = 'cause <(↑ Subj) (↑ Obj) (Comp)>'

Here, a new predicate is formed which causes the action and requires a new subject, while the old subject becomes the object of the new predicate and the old verb becomes the X-complement (complement to infinital VPs).

Long Distance Dependencies and Coordination

In GB, when a category moved, it creates an empty category. In LFG, unbounded movement and coordination is handled by the functional identity and by correlation with the corresponding f-structure. An example will better explain these ideas.

Example 2.7 Consider the wh-movement in the following sentence.

Which picture does Tara like–most?

The f-structure can be represented as follows:

$$\begin{bmatrix} \text{Focus} & \begin{bmatrix} \text{Obl}_{th} & \begin{bmatrix} \text{pred} & \text{'PRO'} \\ \text{Refl} & + \end{bmatrix} \\ \text{Pred} & \text{'picture } \langle(\text{Obl}_{th})\rangle\text{'} \end{bmatrix} \\ \text{Subj} & [\text{pred} \quad \text{'Tara'}] \\ \text{Obj} & [\qquad\qquad] \\ \text{Pred} & \text{'like } \langle(\uparrow\text{Subj})\ (\uparrow\text{Obj})\rangle\text{'} \end{bmatrix}$$

The mechanism of handling these movements and coordination are not detailed here. The only aim is to highlight efforts and issues involved in modelling language.

2.2.5 Paninian Framework

Another very important model which has drawn much attention is the Paninian Grammar-based model (Kiparsky 1982, Bharti et al. 1995). Although *Paninian grammar* (PG) was written by Panini in 500 BC in Sanskrit (the original text being titled *Asthadhyayi*), the framework can be used for other Indian languages and possibly some Asian languages as well.

Unlike English, Asian languages are SOV (Subject-Object-Verb) ordered and inflectionally rich. The inflections provide important syntactic and semantic cues for language analysis and understanding. The Paninian framework takes advantage of these features. However, it should be noted that the research on this framework is still in progress and there are many complexities of Indian languages which are yet to be explained through this or other models. In this section, we briefly discuss some unique features of PG, to provide a glimpse of another potential model.

Some Important Features of Indian Languages

Indian languages have traditionally used oral communication for knowledge propagation. The purpose of these languages is to communicate ideas from the speaker's mind to the listener's mind. Such oral traditions have given rise to a morphologically rich language. Also, they are relatively word-order free. Some languages, like Sanskrit, have the flexibility to allow word groups representing subject, object, and verb to occur in any order. In others, like Hindi, we can change the position of subject and object. For example:

(a) माँ बच्चे को खाना देती है।
Maan Bachche ko khanaa detii hai
Mother child to food give–(s)
Mother gives food to the child.

(b) बच्चे को माँ खाना देती है।
Bachche ko Maan khanaa detii hai
Child to mother food give–(s)
Mother gives food to the child.

The auxilary verbs follow the main verb. In Hindi, they remain as separate words, whereas in south Indian (Dravidian) languages, they combine with the main verb. For example:

खा रहा है	करता रहा है
khaa raha hai	*kartaa rahaa hai*
eat-ing	doing been has
eating	has been doing

In Hindi, some verbs (main), e.g., give (देना), take (लेना), also combine with other verbs (main) to change the aspect and modality of the verbs.

Example 2.8

उसने खाना खाया।	उसने खाना खा लिया।
Usne khanaa khaayaa	*Usne khaanaa kha liyaa*
He (Subj) food ate	He (Subj) food eat taken
He ate food	He ate food (completed the action)
वह चला	वह चल दिया
	He move given
He moved	He moved (started the action)

In Indian languages, the nouns are followed by post-positions instead of prepositions. They generally remain as separate words in Hindi, except in the case of pronouns, for example

रेखा	के	पिता	उसके	पिता
Rekha	ke	pita	*Uske*	*pita*
Rekha	of	father	Her (His) father	
Father	of	Rekha		

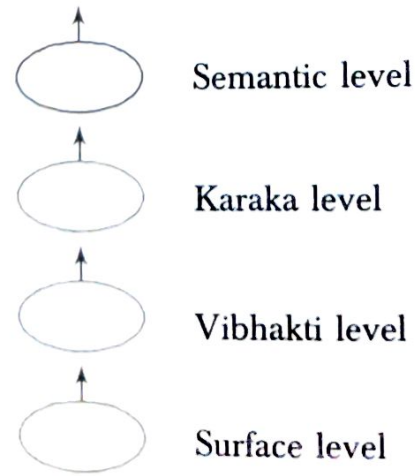
In view of such differences between English (and English-like languages) and Indian languages, it is imperative that we find a new framework for handling Indian languages. Even among Indian languages, all features are not the same. As noted earlier, verb groups are formed differently in Indo-Aryan and Dravidian languages. Sanskrit is very different from the other Indian languages as it has five tenses and three numbers, and only one time aspect in each tense. Hence, the translation of 'He goes' and 'He is going' is the same in Sanskrit. Hindi is unique in the sense that it has no neuter gender. All nouns are categorized as feminine or masculine, and the verb form must have a gender agreement with the subject (sometimes with the object).

Thus, we have

ताला खो गया	चाभी खो गयी
Taalaa kho gayaa	*Chaabhii kho gayeee*
Lock lose (past)	key lose (past)
The lock was lost.	The key was lost.

Layered Representation in PG

The GB theory represents three syntactic levels: deep structure, surface structure, and logical form (LF), where the LF is nearer to semantics. This theory tries to resolve all language issues at syntactic levels only. Unlike GB, Paninian grammar framework is said to be syntactico-semantic, that is, one can go from surface layer to deep semantics by passing through intermediate layers. Although all these layers are not named, as per Bharti, et al. (1995), the language can be represented as follows:



The surface and the semantic levels are obvious. The other two levels should not be confused with the levels of GB. *Vibhakti* literally means inflection, but here, it refers to word (noun, verb, or other) groups based either on case endings, or post-positions, or compound verbs, or main and auxiliary verbs, etc. Instead of talking about NP, VP, AP, PP, etc., word groups are formed based on various kinds of markers (including the absence of it or θ). These markers are language-specific, but all Indian languages (and possibly Asian languages as well) can be represented at the Vibhakti level.

Karaka (pronounced *Kaaraka*) literally means Case, and in GB, we have already discussed case theory, θ-theory, and sub-categorization, etc. Paninian Grammar has its own way of defining Karaka relations, which we discuss in the next section. These relations are based on the way the word groups participate in the activity denoted by the verb group. In this sense, it is semantic as well as syntactic. However, things are not so straightforward. Complexities arise because of the absence of inflections, multiple categories of the words, multiple meanings, and above all, the presence of a large number of exceptions. These exceptions are not only applicable on stated rules but also on future rules. Such forward and backward chaining makes actual implementation difficult.

As the purpose of these languages is to communicate, generally between one human and another, the resolution of ambiguities is a contentious issue, often left to the listener. Hence, there may not be any particular number of semantic levels. Multiple-meaning texts are abundant in Indian literature as seen in the hundreds of interpretations of the epics.

Karaka Theory

Karaka theory is the central theme of PG framework. Karaka relations are assigned based on the roles played by various participants in the main activity. These roles are reflected in the case markers and post-position markers (parsargs). These relations are similar to case relations in English, but the types of relations are defined in a different manner and the richness of the case endings found in Indian languages has been used to its advantage.

We will discuss the various Karakas, such as Karta (subject), Karma (object), Karana (instrument), Sampradana (beneficiary), Apadan (separation), and Adhikaran (locus). These descriptions are just examples and not a complete discussion of PG or Karaka theory. For details of Karaka theory, see Shastri (1973) and Vasu (1977).

To explain various Karaka relations, let us consider an example. Consider the following Hindi sentence:

माँ बच्ची को आँगन में हाथ से रोटी खिलाती है। (2.5)

Maan bachchi ko aangan mein haath se rotii khilaatii hei

Mother child-to courtyard-in hand-by bread feed (s).

The mother feeds bread to the child by hand in the courtyard.

The first important Karak is subject, called 'Karta' in PG. Karta is defined as the noun group which is most independent (*swatantra* in Hindi). Karta has generally 'ne' or 'ϕ' case marker. It is an independent entity in the activity denoted by the main verb. As seen in GB also, verbs do not sub-categorize for subjects, although they assign a θ role to it. In sentence 2.5, 'maan' (mother) is the Karta. The concept of Karta is different from the 'agent' concept in the sense that Karta can also take up the role of experiencer, e.g.,

मुझसे रहा न गया।

Mujhse rahaa na gayaa

Me hold -not -passive

I could not hold myself.

'Karma' is similar to object and is the locus of the result of the activity. In sentence (2.5), *rotii* (bread) is the Karma. As explained earlier, when the Karta is the experiencer, it (she) is also the locus of the result. Thus, the locus of the result is only when it is different from Karta termed Karma. Karma generally has 'ϕ' or 'KO' case marker. Another Karaka relation is 'Karan' (instrument), which is a noun group through which the goal is achieved. In sentence (2.5), *haath* (hand) is the Karan. It has the marker *dwara* (by) or *se*. 'Sampradan' is the beneficiary of the activity, e.g., *bachchi* (child) in sentence (2.5). It takes the marker *ko* (to) or *ke liye* (for). 'Apaadaan' denotes separation and the marker is attached to the part that serves as a reference point (being stationary), for example

माँ ने थाली से खाना उठाकर बच्चे को दिया।

Maan ne thaali se khana uthakar bachche ko diyaa

Mother-Karta plate from Apaadan food taking-up child-to gave.

The mother gave food to the child taking it up from the plate.

Here *thaali* is the Apaadan. 'Adhikaran' is the locus (support in space or time) of Karta or Karma. In sentence (2.5), *aangan* (courtyard) is the Adhikaran. As these six relations are not sufficient to capture all possible relations, various others such as 'Sambandh' (relation) and 'Tadarthya' (purpose) have also been tried.

Issues in Paninian Grammar

The two problems challenging linguists are:

(i) Computational implementation of PG, and
(ii) Adaptation of PG to Indian, and other similar languages.

An approach to implementing PG has been discussed in Bharati, et al. (1995). This is a multilayered implementation. The approach is named 'Utsarga-Apvada' (default-exception), where rules are arranged in multiple layers in such a way that each layer consists of rules which are in exception to rules in the higher layer. Thus, as we go down the layer, more particular information is derived. Rules may be represented in the form of charts (such as Karaka chart and Lakshan chart).

However, many issues remain unresolved, specially in cases of shared Karak relations. Another difficulty arises when mapping between the Vibhakti (case markers and post-positions) and the semantic relation (with respect to verb) is not one to one. Two different Vibhakti can represent the same relation, or the same Vibhakti can represent different relations in different contexts. The strategy to disambiguate the various senses of words, or word groupings, are still the challenging issues.

As the system of rules is different in different languages, the framework requires adaptations to tackle various applications in various languages. Only some general features of PG framework has been described here.

2.3 STATISTICAL LANGUAGE MODEL

A statistical language model is a probability distribution $P(s)$ over all possible word sequences (or any other linguistic unit like words, sentences, paragraphs, documents, or spoken utterances). A number of statistical language models have been proposed in literature. The dominant approach in statistical language modelling is the n-gram model.

2.3.1 *n*-gram Model

As discussed earlier, the goal of a statistical language model is to estimate the probability (likelihood) of a sentence. This is achieved by decomposing sentence probability into a product of conditional probabilities using the chain rule as follows:

$$\begin{aligned} P(s) &= P(w_1, w, w_3, \ldots, w_n) \\ &= P(w_1)\, P(w_2/w_1)\, P(w_3/w_1\, w_2)\, P(w_4/w_1\, w_2\, w_3) \ldots \\ &\qquad P(w_n/w_1\, w_2 \ldots w_{n-1})) \\ &= \prod_{i=1}^{n} P(w_i/h_i) \end{aligned}$$

where h_i is history of word w_i defined as

$$w_1 w_2 \ldots w_{i-1}$$

So, in order to calculate sentence probability, we need to calculate the probability of a word, given the sequence of words preceding it. This is not a simple task. An *n*-gram model simplifies the task by approximating the probability of a word given all the previous words by the conditional probability given previous *n*-1 words only.

$$P(w_i/h_i) \approx P(w_i/w_{i-n+1} \ldots w_{i-1})$$

Thus, an *n*-gram model calculates $P(w_i/h_i)$ by modelling language as Markov model of order $n-1$, i.e., by looking at previous *n*-1 words only. A model that limits the history to the previous one word only, is termed a bi-gram ($n = 1$) model. Likewise, a model that conditions the probability of a word to the previous two words, is called a tri-gram ($n = 2$) model. Using bi-gram and tri-gram estimate, the probability of a sentence can be calculated as:

$$P(s) \approx \prod_{i=1}^{n} P(w_i/w_{i-1})$$

and $$P(s) \approx \prod_{i=1}^{n} P(w_i/w_{i-2} \cdot w_{i-1})$$

As an example, the bi-gram approximation of *P*(east/The Arabian knights are fairy tales of the) is

P(east/the),

whereas a tri-gram approximation is

P(east/of the).

A special word (pseudo word) <s> is introduced to mark the beginning of the sentence in bi-gram estimation. The probability of the first word in a sentence is conditioned on <s>. Similarly, in tri-gram estimation, we introduce two pseudo-words <s1> and <s2>.

Now, we discuss how to estimate these probabilities. This is done by training the *n*-gram model on the training corpus. We estimate *n*-gram parameters using the maximum likelihood estimation (MLE) technique, i.e., using relative frequencies. We count a particular *n*-gram in the training corpus and divide it by the sum of all *n*-grams that share the same prefix.

$$P(w_i/w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{\sum_{w} C(w_{i-n+1}, \ldots, w_{i-1}, w)}$$

The sum of all n-grams that share first $n-1$ words is equal to the count of the common prefix $w_{i-n+1}, \ldots, w_{i-1}$. So, we rewrite the previous expression as follows:

$$P(w_i/w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1})}$$

The model parameter we get using these estimates, maximizes the probability of the training set T given the model M, i.e., $P(T/M)$. The frequency with which a word occurs in a text may not be the same as in the training set; this model only provides the most likely solution.

A number of improvements have been suggested for the standard n-gram model. Before we discuss them, let us illustrate these ideas with the help of an example.

Example 2.9

Training set:

The Arabian Knights
These are the fairy tales of the east
The stories of the Arabian knights are translated in many languages

Bi-gram model:

P(the/<s>) = 0.67	P(Arabian/the) = 0.4	P(knights /Arabian) = 1.0
P(are/these) = 1.0	P(the/are) = 0.5	P(fairy/the) = 0.2
P(tales/fairy) = 1.0	P(of/tales) = 1.0	P(the/of) = 1.0
P(east/the) = 0.2	P(stories/the) = 0.2	P(of/stories) = 1.0
P(are/knights) = 1.0	P(translated/are) = 0.5	P(in /translated) = 1.0
P(many/in) = 1.0		
P(languages/many) = 1.0		

Test sentence(s): The Arabian knights are the fairy tales of the east.

P(The/<s>) × P(Arabian/the) × P(Knights/Arabian) × P(are/knights) × P(the/are) × P(fairy/the) × P(tales/fairy) × P(of/tales) × P(the/of) × P(east/the)

$= 0.67 \times 0.5 \times 1.0 \times 1.0 \times 0.5 \times 0.2 \times 1.0 \times 1.0 \times 1.0 \times 0.2$

$= 0.0067$

As each probability is necessarily less than 1, multiplying the probabilities might cause a numerical underflow, particularly in long sentences. To avoid this, calculations are made in log space, where a calculation corresponds to adding log of individual probabilities and taking antilog of the sum.

The n-gram model suffers from data sparseness problem. An n-gram that does not occur in the training data is assigned zero probability, so that even a large corpus has several zero entries in its bi-gram matrix. This is because of the assumption that the probability of occurrence of a word depends only on the preceding word (or preceding n–1 words), which is not true in general. There are several long distance dependencies in natural language sentences, which this model fails to capture. Goodman (2003) pointed out that 'there is rarely enough data to accurately estimate the parameters of a language model.'

A number of smoothing techniques have been developed to handle the data sparseness problem, the simplest of these being add-one smoothing. In the words of Jurafsky and Martin (2000):

> *Smoothing in general refers to the task of re-evaluating zero-probability or low-probability n-grams and assigning them non-zero values.*

The word 'smoothing' is used to denote these techniques because they tend to make distributions more uniform by moving the extreme probabilities towards the average.

2.3.2 Add-one Smoothing

This is the simplest smoothing technique. It adds a value of one to each n-gram frequency before normalizing them into probabilities. Thus, the conditional probability becomes:

$$P(w_i/w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1}) + V}$$

where V is the vocabulary size, i.e., size of the set of all the words being considered.

In general, add-one smoothing is not considered a good smoothing technique. It assigns the same probability to all missing n-grams, even though some of them could be more intuitively appealing than others. Gale and Church (1994) reported that variance of the counts produced by the add-one smoothing is worse than the unsmoothed MLE method. Another problem with this technique is that it shifts too much of the probability mass towards the unseen n-grams (n-grams with 0 probabilities) as there number is usually quite large. Good–Turing smoothing (Good 1953) attempts to improve the situation by looking at the number of n-grams with a high frequency in order to estimate the probability mass that needs to be assigned to missing or low-frequency n-grams.

2.3.3 Good-Turing Smoothing

Good-Turing smoothing (Good 1953) adjusts the frequency f of an n-gram using the count of n-grams having a frequency of occurrence $f+1$. It converts the frequency of an n-gram from f to f^* using the following expression:

$$f^* = (f+1)\frac{n_{f+1}}{n_f}$$

where n_f is the number of n-grams that occur exactly f times in the training corpus. As an example, consider that the number of n-grams that occur 4 times is 25,108 and the number of n-grams that occur 5 times is 20,542. Then, the smoothed count for 5 will be

$$\frac{20542}{25108} \times 5 = 4.09$$

2.3.4 Caching Technique

Another improvement over basic n-gram model is caching. The frequency of n-gram is not uniform across the text segments or corpus. Certain words occur more frequently in certain segments (or documents) and rarely in others. For example, in this section, the frequency of the word 'n-gram' is high, whereas it occurs rarely in earlier sections. The basic n-gram model ignores this sort of variation of n-gram frequency. The cache model combines the most recent n-gram frequency with the standard n-gram model to improve its performance locally. The underlying assumption here is that the recently discovered words are more likely to be repeated.

A number of other smoothing techniques appear in literature. For these, readers are referred to Chen and Goodman (1998).

SUMMARY

The main topics covered in this chapter are as follows.

- Language modelling deals with providing a description of natural languages amenable to processing.
- There are two main approaches to language modelling: grammar-based language model and statistical language model.
- A grammar-based language model uses grammar to model language. A number of computational grammars have been proposed in literature. This chapter provides a discussion of models based on

government and binding, lexical functional grammar, and Paninian grammar.

- GB theory talks about representations at four different levels: s-structure, d-structure, logical form, and phonetic form.
- An important concept of GB is a general transformational rule called Move α which moves constituents freely, subject to certain constraints (defined by several theories and principles).
- LFG represents sentences at two different levels—constituent structure (c-structure) and functional structure (f-structure).
- Paninian grammar provides a framework for modelling Indian languages.
- The Paninian framework is syntactico-semantic. The central theme of this framework is Karaka relations.
- A statistical language model estimates the probability (likelihood) of a sentence. The most widely used statistical model is n-gram model.
- The *n*-gram model suffers from sparseness of data. Smoothing techniques such as add-one and Good–Turing can be used to handle this problem.
- Caching is another improvement over the standard *n*-gram model.

REFERENCES

Bharati, A., V. Chaitanya, and R. Sangal, 1995, 'Natural Language Processing: A Paninian Perspective,' Prentice-Hall of India, New Delhi.

Bresnan, Joan, 1976, 'Evidence for a theory of unbounded transformation,' *Linguistic Analysis.*

_____1977, 'Variables in theory of transformations,' *Formal Syntax*, Peter W. Culicover, Thoman Wasow, and Adrian Akmajian (Eds.), Academic Press, New York, pp. 157–96.

Chen, Stanley F. and Joshua T. Goodman, 1998, 'An Empirical Study of Smoothing Techniques for Language Modelling,' *Technical Report TR-10–98*, Computer Science Group, Harvard University.

Chomsky, Noam, 1957, *Syntactic Structures*, Mouton, The Hague.

_____1986, *Some Concepts and Consequences of the Theory of Government and Binding*, MIT Press, Cambridge, MA.

Gale, William A. and Kenneth W. Church, 1994, 'What's wrong with adding one?,' Corpus-based Research into Language, Oostdijk and P. de Haan (Eds.), Rodolpi, Amsterdam.

Gazdar, G., E. Klein, G. Pullum, and I. Sag, 1985, *Generalized Phrase Structure Grammar*, Blackwell.

LIBRARY
Acc. No. 162201
Date

Good, I.J., 1953, 'The population frequencies of species and the estimation of population parameters,' *Biometrika*, 40(3 and 4), pp. 237–64.

Goodman, Joshua, 2003, 'The state of the art in language modelling,' *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials*, 5, Edmonton, Canada.

Jurafsky, Daniel and James H. Martin, 2000, 'Speech and Language Processing: An Introduction to Natural Language Processing,' *Computational Linguistics and Speech Recognition*, Prentice Hall, New Jersey.

Kaplan, Ronald M., 1975a, 'On Process Models for Sentence Comprehension,' *Explorations in Cognition*, Donald A. Norman and David E. Rumelhart (Eds.), W.H. Freeman, San Francisco.

_____1975b, 'Transient Processing Load in Relative Clauses,' *Doctoral Dissertation*, Harvard University.

_____1995, 'The Formal Architecture of Lexical-Functional Grammar,' *Formal Issues in Lexical-Functional Grammar*, M. Dalrymple, R.M. Kaplan, J.T. Maxwell III, and A. Zaenen (Eds.), CSLI Publications, Stanford.

Kaplan, Ronald M. and Joan Bresnan, 1982, *The Mental Representation of Grammatical Relations*, Joan Bresnan (Ed.), The MIT Press, Cambridge, MA.

Kiprasky, P., 1982, 'Some theoretical problem in Panini's grammar,' Bhandarkar Oriental Research Institute, Poona.

Rosenfeld, Ronald, 1994, 'Adaptive Statistical Language Modelling: A Maximum Entropy Approach,' *D.Phil. Thesis*, Carnegie Mellon University, Pittsburgh.

Sells, Peter, 1985, 'Lectures on Contemporary Syntactic Theories,' Center for the Study of Language and Information (CSLI), *Lecture Notes*, Number 3, Stanford.

Shastri, Charudev, 1973, *Vyakarana Chandrodaya*, (Vol. I–V), Motilal Banarsidass, in Hindi, New Delhi

Vasu, S.C. (Tr.), 1977, *The Ashtadhyayi of Panini* (2 volumes), Motilal Banarsidass, New Delhi.

Woods, William A., 1970, 'Transition Network Grammars for Natural Language Analysis,' *Communications of the ACM*, 13(10), pp. 591–606.

EXERCISES

1. Give the representation of a sentence in d-structure and s-structure in GB.
2. How is the structure of a sentence different from the structure of a phrase?

3. Discuss empty-category principle and give two examples of the creation of empty categories.
4. What is f-structure in LFG? What inputs to an algorithm create an f structure?
5. Using the annotated rules in Example 2.5, find f-structures for the various phrases and the complete sentence, 'I saw Aparna in the market at night'.
6. What are Karaka relations? Explain the difference between Karta and Agent.
7. What are lexical rules? Give the complete entry for a verb in the lexicon, to be used in LFG.
8. Compare GB and PG. Why is PG called syntactico-semantic theory?
9. What are the problems associated with n-gram model? How are these problems handled?
10. How does caching improve the *n*-gram model?

LAB EXERCISES

1. Create a collection of 10 documents. Write a program to find the frequency of bi-grams in this collection.
2. Find the number of bi-grams that occur more than three times in this collection.