Natural Language Processing
Unit-I

Dr A Nagesh

Unit-I

1.Finding the Structure of Words

This section deals with words, its structure and its models

1.1 Words and Their Components

- 1.1.1 Tokens
- 1.1.2 Lexemes
- 1.1.3 Morphemes
- 1.1.4 Typology

1.2 Issues and Challenges

- 1.2.1 Irregularity
- 1.2.2 Ambiguity
- 1.2.3 Productivity

1.3 Morphological Models

- 1.3.1 Dictionary Lookup
- 1.3.2 Finite-State Morphology
- 1.3.3 Unification-Based Morphology
- 1.3.4 Functional Morphology
- 1.3.5 Morphology Induction

2.**Finding the Structure of Documents**

This chapter mainly deals with Sentence and topic detection or segmentation.

2.1 Introduction

- 2.1.1 Sentence Boundary Detection
- 2.1.2 Topic Boundary Detection

2.2 Methods

This section deals with statistical classical approaches (Generative and Discriminative approaches)

- 2.2.1 Generative Sequence Classification Methods
- 2.2.2 Discriminative Local Classification Methods
- 2.3.3 Discriminative Sequence Classification Methods
- 2.2.4 Hybrid Approaches
- 2.2.5 Extensions for Global Modelling for Sentence Segmentation

2.3 Complexity of the Approaches

2.4 Performance of the Approaches

NATURAL LANGUAGE PROCESSING(NLP)

UNIT - I

i.Finding the Structure of Words:

- Words and Their Components
- Issues and Challenges
- Morphological Models

ii.Finding the Structure of Documents:

- Introduction
- Methods
- Complexity of the Approaches
- Performances of the Approaches

Natural Language Processing

- Humans communicate through some form of language either by text or speech.
- To make interactions between computers and humans, computers need to understand natural languages used by humans.
- Natural language processing is all about making computers learn, understand, analyse, manipulate and interpret natural(human) languages.
- NLP stands for **Natural Language Processing**, which is a part of **Computer Science, Human language,** and **Artificial Intelligence**.
- Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.
- The ability of machines to interpret human language is now at the core of many applications that we use every day - chatbots, Email classification and spam filters, search engines, grammar checkers, voice assistants, and social language translators.
- The input and output of an NLP system can be Speech or Written Text

Components of NLP

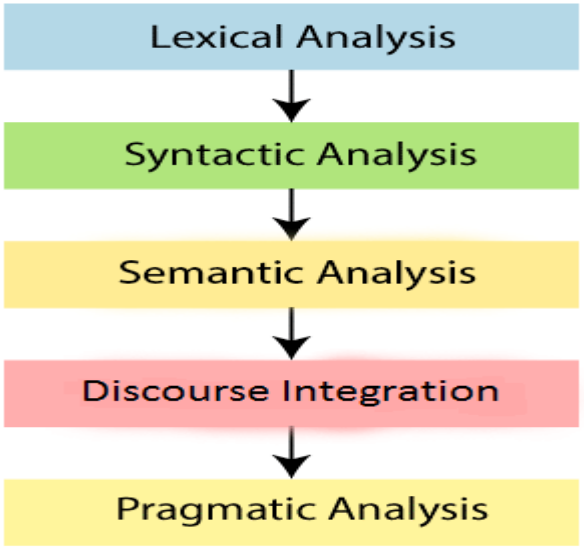
- **There are two components of NLP, Natural Language Understanding (NLU) and Natural Language Generation (NLG).**
- Natural Language Understanding (NLU) which involves transforming human language into a machine-readable format.
- It helps the machine to understand and analyse human language by extracting the text from large data such as keywords, emotions, relations, and semantics.
- Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation.
- It mainly involves Text planning, Sentence planning, and Text realization.
- The NLU is harder than NLG.

NLP Terminology

- **Phonology** − It is study of organizing sound systematically.
- ***Morphology***: The study of the formation and internal structure of words.
- **Morpheme** − It is primitive unit of meaning in a language.
- ***Syntax***: The study of the formation and internal structure of sentences.
- ***Semantics***: The study of the meaning of sentences.
- **Pragmatics** − It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected.
- **Discourse** − It deals with how the immediately preceding sentence can affect the interpretation of the next sentence.
- **World Knowledge** − It includes the general knowledge about the world.

Steps in NLP

- There are general five steps :
    1. Lexical Analysis
    2. Syntactic Analysis (Parsing)
    3. Semantic Analysis
    4. Discourse Integration
    5. Pragmatic Analysis



Lexical Analysis –

- The first phase of NLP is the Lexical Analysis.
- This phase scans the source code as a stream of characters and converts it into meaningful lexemes.
- It divides the whole text into paragraphs, sentences, and words.

Syntactic Analysis (Parsing) –

- Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.
- The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.

Semantic Analysis –

- Semantic analysis is concerned with the meaning representation.
- It mainly focuses on the literal meaning of words, phrases, and sentences.
- The semantic analyzer disregards sentence such as “hot ice-cream”.

Discourse Integration –

- Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

Pragmatic Analysis –

- During this, what was said is re-interpreted on what it actually meant.
- It involves deriving those aspects of language which require real world knowledge.
- **Example:** "Open the door" is interpreted as a request instead of an order.

Finding the Structure of Words

- Human language is a complicated thing.
- We use it to express our thoughts, and through language, we receive information and infer its meaning.
- Trying to understand language all together is not a viable approach.
- Linguists have developed whole disciplines that look at language from different perspectives and at different levels of detail.
- The point of **morphology**, for instance, is to study the variable forms and functions of words,
- The syntax is concerned with the arrangement of words into phrases, clauses, and sentences.
- Word structure constraints due to pronunciation are described by **phonology**,
- The conventions for writing constitute the **orthography** of a language.
- The meaning of a linguistic expression is its semantics, and etymology and lexicology cover especially the evolution of words and explain the semantic, morphological, and other links among them.
- Words are perhaps the most intuitive units of language, yet they are in general tricky to define.
- Knowing how to work with them allows, in particular, the development of **syntactic** and **semantic** abstractions and simplifies other advanced views on language.
- Here, first we explore how to identify words of distinct types in human languages, and how the internal structure of words can be modelled in connection with the grammatical properties and lexical concepts the words should represent.

- The discovery of word structure is **morphological parsing**.
- In many languages, words are delimited in the orthography by whitespace and punctuation.
- But in many other languages, the writing system leaves it up to the reader to tell words apart or determine their exact phonological forms.

Words and Their Components

- Words are defined in most languages as the smallest linguistic units that can form a complete utterance by themselves.
- The minimal parts of words that deliver aspects of meaning to them are called **morphemes**.

Tokens

- Suppose, for a moment, that words in English are delimited only by whitespace and punctuation (the marks, such as full stop, comma, and brackets)
- Example: Will you read the newspaper? Will you read it? I won't read it.

- If we confront our assumption with insights from syntax, we notice two here: words ***newspaper*** and ***won't***.
- Being a compound word, ***newspaper*** has an interesting **derivational structure**.
- In writing, *newspaper* and the associated concept is distinguished from the isolated *news* and *paper*.
- For reasons of generality, linguists prefer to analyze *won't* as two syntactic words, or tokens, each of which has its independent role and can be reverted to its normalized form.
- The structure of ***won't*** could be parsed as ***will*** followed by ***not***.
- In English, this kind of tokenization and **normalization** may apply to just a limited set of cases, but in other languages, these phenomena have to be treated in a less trivial manner.
- In Arabic or Hebrew, certain tokens are concatenated in writing with the preceding or the following ones, possibly changing their forms as well.
- The underlying lexical or syntactic units are thereby blurred into one compact string of letters and no longer appear as distinct words.
- Tokens behaving in this way can be found in various languages and are often called **clitics**.
- In the writing systems of Chinese, Japanese, and Thai, whitespace is not used to separate words.

Lexemes

- By the term word, we often denote not just the one **linguistic form** in the given context but also the **concept behind the form** and the **set of alternative forms** that can express it.
- Such **sets** are called **lexemes or lexical items**, and they constitute the **lexicon** of a language.
- Lexemes can be divided by their behaviour into the lexical categories of verbs, nouns, adjectives, conjunctions, particles, or other parts of speech.
- The citation **form of a lexeme**, by which it is commonly identified, is also called its **lemma**.
- When we convert a word into its other forms, such as turning the **singular *mouse*** into the **plural *mice* or *mouses***, we say we **inflect** the lexeme.
- When we transform a lexeme into another one that is morphologically related, regardless of its lexical category, we say we derive the lexeme: for instance, the nouns ***receiver* and *reception*** are derived from the verb *to* ***receive***.
- Example: Did you see him? I **didn’t** see him. I didn’t see **anyone**.
- Example presents the problem of tokenization of ***didn’t*** and the investigation of the internal structure of ***anyone***.

- In the paraphrase ***I saw no one***, the lexeme ***to see*** would be inflected into the form ***saw*** to reflect its grammatical function of expressing **positive past tense**.
- Likewise, ***him*** is the oblique case form of ***he*** or even of a more abstract lexeme representing all personal pronouns.
- In the paraphrase, ***no one*** can be perceived as the minimal word synonymous with ***nobody***.
- The difficulty with the definition of what counts as a word need not pose a problem for the syntactic description if we understand ***no one*** **as two closely connected tokens treated as one fixed element**.

Morphemes

- Morphological theories differ on whether and how to associate the properties of word forms with their structural components.
- These components are usually called **segments** or **morphs**.
- The morphs that by themselves represent some aspect of the meaning of a word are called **morphemes** of some function.
- Human languages employ a variety of devices by which morphs and morphemes are combined into word forms.

Morphology

- Morphology is the domain of linguistics that analyses the internal structure of words.
- Morphological analysis – exploring the structure of words
- Words are built up of minimal meaningful elements called morphemes:
  - played = play-ed
  - cats = cat-s
  - unfriendly = un-friend-ly
- Two types of morphemes:
  - i Stems: play, cat, friend
  - ii Affixes: -ed, -s, un-, -ly
- Two main types of affixes:
  - i Prefixes precede the stem: un-
  - ii Suffixes follow the stem: -ed, -s, un-, -ly
- Stemming = find the stem by stripping off affixes
  - play = play
  - replayed = re-play-ed
  - computerized = comput-er-ize-d

Problems in morphological processing

- Inflectional morphology: inflected forms are constructed from base forms and inflectional affixes.
- Inflection relates different forms of the same word

Lemma	Singular	Plural
cat	cat	cats
dog	dog	dogs
knife	knife	knives
sheep	sheep	sheep
mouse	mouse	mice

- Derivational morphology: words are constructed from roots (or stems) and derivational affixes:

  inter+national = international

  international+ize = internationalize

  internationalize+ation = internationalization

- The simplest morphological process concatenates morphs one by one, as in *dis-agree-ment-s*, where *agree* is a free lexical morpheme and the other elements are bound grammatical morphemes contributing some partial meaning to the whole word.
- in a more complex scheme, morphs can interact with each other, and their forms may become subject to additional phonological and orthographic changes denoted as morphophonemic.
- The alternative forms of a morpheme are termed **allomorphs**.

Typology

- Morphological typology divides languages into groups by characterizing the prevalent morphological phenomena in those languages.
- It can consider various criteria, and during the history of linguistics, different classifications have been proposed.
- Let us outline the typology that is based on quantitative relations between words, their morphemes, and their features:
- **Isolating**, or **analytic**, languages include no or relatively few words that would comprise more than one morpheme (typical members are Chinese, Vietnamese, and Thai; analytic tendencies are also found in English).

- **Synthetic** languages can combine more morphemes in one word and are further divided into agglutinative and fusional languages.
- **Agglutinative** languages have morphemes associated with only a single function at a time (as in Korean, Japanese, Finnish, and Tamil, etc.)
- **Fusional** languages are defined by their feature-per-morpheme ratio higher than one (as in Arabic, Czech, Latin, Sanskrit, German, etc.).
- In accordance with the notions about word formation processes mentioned earlier, we can also find out using concatenative and nonlinear:
- **Concatenative** languages linking morphs and morphemes one after another.
- **Nonlinear** languages allowing structural components to merge nonsequentially to apply tonal morphemes or change the consonantal or vocalic templates of words.

Morphological Typology

- **Morphological typology** is a way of classifying the languages of the world that groups languages according to their common morphological structures.
- The field organizes languages on the basis of how those languages form words by combining morphemes.
- The morphological typology classifies languages into **two broad classes** of **synthetic languages** and **analytical languages.**
- The **synthetic class** is then further sub classified as either **agglutinative languages** or **fusional languages.**
- Analytic languages contain very little inflection, instead relying on features like word order and auxiliary words to convey meaning.
- Synthetic languages, ones that are not analytic, are divided into two categories: agglutinative and fusional languages.
- Agglutinative languages rely primarily on discrete particles(prefixes, suffixes, and infixes) for inflection, ex: inter+national = international, international+ize = internationalize.
- While fusional languages "fuse" inflectional categories together, often allowing one word ending to contain several categories, such that the original root can be difficult to extract (anybody, newspaper).

Issues and Challenges

- **Irregularity**: word forms are not described by a prototypical linguistic model.
- **Ambiguity**: word forms be understood in multiple ways out of the context of their discourse.
- **Productivity**: is the inventory of words in a language finite, or is it unlimited?
- Morphological parsing tries to eliminate the variability of word forms to provide higher-level linguistic units whose lexical and morphological properties are explicit and well defined.
- It attempts to remove unnecessary irregularity and give limits to ambiguity, both of which are present inherently in human language.
- By irregularity, we mean existence of such forms and structures that are not described appropriately by a prototypical linguistic model.
- Some irregularities can be understood by redesigning the model and improving its rules, but other lexically dependent irregularities often cannot be generalized

- Ambiguity is indeterminacy (not being interpreted) in interpretation of expressions of language.
- Morphological modelling also faces the problem of productivity and creativity in language, by which unconventional but perfectly meaningful new words or new senses are coined.

Irregularity

- Morphological parsing is motivated by the quest for generalization and abstraction in the world of words.
- Immediate descriptions of given linguistic data may not be the ultimate ones, due to either their inadequate accuracy or inappropriate complexity, and better formulations may be needed.
- The design principles of the morphological model are therefore very important.
- In Arabic, the deeper study of the morphological processes that are in effect during inflection and derivation, even for the so-called irregular words, is essential for mastering the whole morphological and phonological system.
- With the proper abstractions made, irregular morphology can be seen as merely enforcing some extended rules, the nature of which is phonological, over the underlying or prototypical regular word forms.

P-STEM	P—3MS	P—2FS	P—3MP	II2MS	IS1—S	IJ1—S	I-STEM	
qaraʾ	*qaraʾa*	*qaraʾti*	*qaraʾū*	*taqraʾu*	*ʾaqraʾa*	*ʾaqraʾ*	*qraʾ*	S
faʿal	*faʿal-a*	*faʿal-ti*	*faʿal-ū*	*ta-fʿal-u*	*ʾa-fʿal-a*	*ʾa-fʿal*	*fʿal*	I
faʿal	*faʿal-a*	*faʿal-ti*	*faʿal-ū*	*ta-fʿal-u*	*ʾa-fʿal-a*	*ʾa-fʿal-*	*fʿal*	M
...	...-*a*	...-*ti*	...-*ū*	*ta-...-u*	*ʾa-...-a*	*ʾa-...-*	...	
faʿā	*faʿā-a*	*faʿā-ti*	*faʿā-ū*	*ta-fā-u*	*ʾa-fā-a*	*ʾa-fā-*	*fā*	M
faʿā	*faʿā*	*faʿal-ti*	*faʿ-aw*	*ta-fā*	*ʾa-fā*	*ʾa-fa*	*fā*	I
raʾā	*raʾā*	*raʾayti*	*raʾaw*	*tarā*	*ʾarā*	*ʾara*	*rā*	S

Table: Discovering the regularity of Arabic morphology using morphophonemic templates, where uniform structural operations apply to different kinds of stems.

In rows, surface forms S of *qara_* ‘to read’ and *ra_ ‿a* ‘to see’ and their inflections are analyzed into immediate I and morphophonemic M templates, in which dashes mark the structural boundaries where merge rules are enforced.

The outer columns of the table correspond to P perfective and I imperfective stems declared in the lexicon; the inner columns treat active verb forms of the following morphosyntactic properties: I indicative, S subjunctive, J jussive mood; 1 first, 2 second, 3 third person; M masculine, F feminine gender; S singular, P plural number.

- Table illustrates differences between a naive model of word structure in Arabic and the model proposed in Smrˇz and Smrˇz and Bielick´y where morphophonemic merge rules and templates are involved.

- Morphophonemic templates capture morphological processes by just organizing stem patterns and generic affixes without any context-dependent variation of the affixes or ad hoc modification of the stems.
- The merge rules, indeed very neatly or effectively concise, then ensure that such structured representations can be converted into exactly the surface forms, both orthographic and phonological, used in the natural language.
- Applying the merge rules is independent of and irrespective of any grammatical parameters or information other than that contained in a template.
- Most morphological irregularities are thus successfully removed.

Ambiguity

- Morphological ambiguity is the possibility that word forms be understood in multiple ways out of the context of their discourse (communication in speech or writing).
- Words forms that look the same but have distinct functions or meaning are called homonyms.
- Ambiguity is present in all aspects of morphological processing and language processing at large.

- Table arranges homonyms on the basis of their behaviour with different endings.

Systematic homonyms arise as verbs combined with endings in Korean

(-ko)		(-e)		(-un)		Meaning
묻고	*mwut.ko*	묻어	*mwut.e*	묻은	*mwut.un*	'bury'
묻고	*mwut.ko*	물어	*mwul.e*	물은	*mwul.un*	'ask'
물고	*mwul.ko*	물어	*mwul.e*	문	*mwun*	'bite'
걷고	*ket.ko*	걷어	*ket.e*	걷은	*ket.un*	'roll up'
걷고	*ket.ko*	걸어	*kel.e*	걸은	*kel.un*	'walk'
걸고	*kel.ko*	걸어	*kel.e*	건	*ken*	'hang'
굽고	*kwup.ko*	굽어	*kwup.e*	굽은	*kwup.un*	'be bent'
굽고	*kwup.ko*	구워	*kwu.we*	구운	*kwu.wun*	'bake'
이르고	*i.lu.ko*	이르러	*i.lu.le*	이른	*i.lun*	'reach'
이르고	*i.lu.ko*	일러	*il.le*	이른	*i.lun*	'say'

- Arabic is a language of rich morphology, both derivational and inflectional.
- Because Arabic script usually does not encode short vowels and omits yet some other diacritical marks that would record the phonological form exactly, the degree of its morphological ambiguity is considerably increased.
- When inflected syntactic words are combined in an utterance, additional phonological and orthographic changes can take place, as shown in Figure.
- In Sanskrit, one such euphony rule is known as external *sandhi.*

dirāsatī	دراستي	drAsty	→	*dirāsatu ī*	دراسة ي	drAsp y
			→	*dirāsati ī*	دراسة ي	drAsp y
			→	*dirāsata ī*	دراسة ي	drAsp y
muʿallimīya	معلمي	mElmy	→	*muʿallimū ī*	معلمو ي	mElmw y
			→	*muʿallimī ī*	معلمي ي	mElmy y
katabtumūhā	كتبتموها	ktbtmwhA	→	*katabtum hā*	كتبتم ها	ktbtm hA
ʾiğrāʾuhu	إجراؤه	IjrAWh	→	*ʾiğrāʾu hu*	إجراء ه	IjrA’ h
ʾiğrāʾihi	إجرائه	IjrA}h	→	*ʾiğrāʾi hu*	إجراء ه	IjrA’ h
ʾiğrāʾahu	إجراءه	IjrA’h	→	*ʾiğrāʾa hu*	إجراء ه	IjrA’ h
li-ʾl-ʾasafi	للأسف	llOsf	→	*li ʾl-ʾasafi li*	ل الأسف	l AlOsf

- cases are expressed by the same word form with *dirāsatī* ‘my study’ and *muʿallimīya* ‘ my teachers’, but the original case endings are distinct.

Productivity

- Is the inventory of words in a language finite, or is it unlimited?
- This question leads directly to discerning two fundamental approaches to language, summarized in the distinction between *langue* and *parole*, or in the competence versus performance duality by Noam Chomsky.
- In one view, language can be seen as simply a collection of utterances (parole) actually pronounced or written (performance).
- This ideal data set can in practice be approximated by linguistic corpora, which are finite collections of linguistic data that are studied with empirical(based on) methods and can be used for comparison when linguistic models are developed.
- Yet, if we consider language as a system (langue), we discover in it structural devices like recursion, iteration, or compounding(make up; constitute)that allow to produce (competence) an infinite set of concrete linguistic utterances.
- This general potential holds for morphological processes as well and is called morphological productivity.
- We denote the set of word forms found in a corpus of a language as its vocabulary.

- The members of this set are word types, whereas every original instance of a word form is a word token.
- The distribution of words or other elements of language follows the “80/20 rule,” also known as the law of the vital few.
- It says that most of the word tokens in a given corpus can be identified with just a couple of word types in its vocabulary, and words from the rest of the vocabulary occur much less commonly if not rarely in the corpus.
- Furthermore, new, unexpected words will always appear as the collection of linguistic data is enlarged.
- In Czech, negation is a productive morphological operation. Verbs, nouns, adjectives, and adverbs can be prefixed with *ne-* to define the complementary lexical concept.

Morphological Models

- There are many possible approaches to designing and implementing morphological models.
- Over time, computational linguistics has witnessed the development of a number of formalisms and frameworks, in particular grammars of different kinds and expressive power, with which to address whole classes of problems in processing natural as well as formal languages.
- Let us now look at the most prominent types of computational approaches to morphology.

Dictionary Lookup

- Morphological parsing is a process by which word forms of a language are associated with corresponding linguistic descriptions.
- Morphological systems that specify these associations by merely enumerating(is **the act or process of making or stating a list of things one after another)** them case by case do not offer any generalization means.
- Likewise for systems in which analyzing a word form is reduced to looking it up verbatim in word lists, dictionaries, or databases, unless they are constructed by and kept in sync with more sophisticated models of the language.

- In this context, a dictionary is understood as a data structure that directly enables obtaining some precomputed results, in our case word analyses.
- The data structure can be optimized for efficient lookup, and the results can be shared. Lookup operations are relatively simple and usually quick.
- Dictionaries can be implemented, for instance, as lists, binary search trees, tries, hash tables, and so on.
- Because the set of associations between word forms and their desired descriptions is declared by plain enumeration, the coverage of the model is finite and the generative potential of the language is not exploited.
- Despite all that, an enumerative model is often sufficient for the given purpose, deals easily with exceptions, and can implement even complex morphology.
- For instance, dictionary-based *approaches* to Korean depend on a large dictionary of all possible combinations of allomorphs and morphological alternations.
- These approaches do not allow development of reusable morphological rules, though.

Finite-State Morphology

- By finite-state morphological models, we mean those in which the specifications written by human programmers are directly compiled into finite-state transducers.
- The two most popular tools supporting this approach, XFST (Xerox Finite-State Tool) and LexTools.
- Finite-state transducers are computational devices extending the power of finite-state automata.
- They consist of a finite set of nodes connected by directed edges labeled with pairs of input and output symbols.
- In such a network or graph, nodes are also called states, while edges are called arcs.
- Traversing the network from the set of initial states to the set of final states along the arcs is equivalent to reading the sequences of encountered input symbols and writing the sequences of corresponding output symbols.
- The set of possible sequences accepted by the transducer defines the input language; the set of possible sequences emitted by the transducer defines the output language.

Input	Input Morphological parsed output
Cats	cat +N +PL
Cat	cat +N +SG
Cities	city +N +PL
Geese	goose +N +PL
Goose	goose +N +SG) or (goose +V)
Gooses	goose +V +3SG
merging	merge +V +PRES-PART
Caught	(caught +V +PAST-PART) or (catch +V +PAST)

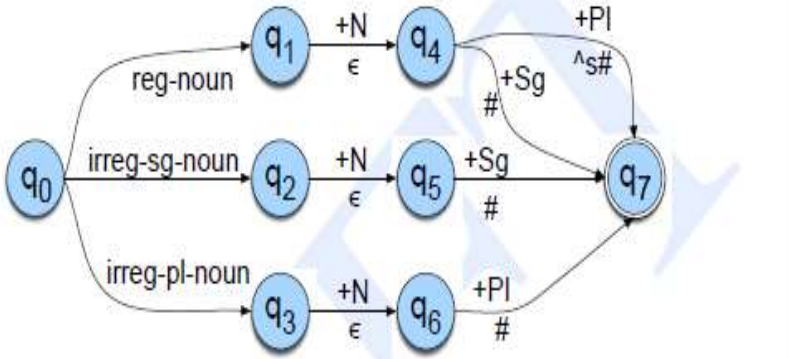


Figure 3.13 A schematic transducer for English nominal number inflection T_{num}. The symbols above each arc represent elements of the morphological parse in the lexical tape; the symbols below each arc represent the surface tape (or the intermediate tape, to be described later), using the morpheme-boundary symbol ^ and word-boundary marker #. The labels on the arcs leaving q_0 are schematic, and need to be expanded by individual words in the lexicon

- For example, a finite-state transducer could translate the infinite regular language consisting of the words *vnuk*, *pravnuk*, *prapravnuk*, *...* to the matching words in the infinite regular language defined by *grandson*, *great-grandson*, *great-great-grandson*.
- In finite-state computational morphology, it is common to refer to the input word forms as **surface strings** and to the output descriptions as **lexical strings**, if the transducer is used for morphological analysis, or vice versa, if it is used for morphological generation.

•In English, a finite-state transducer could analyze the surface string children into the lexical string child [+plural], for instance, or generate women from woman [+plural].

- Relations on languages can also be viewed as functions. Let us have a relation *R*, and let us denote by [Σ] the set of all sequences over some set of symbols Σ, so that the domain and the range of *R* are subsets of [Σ].
- We can then consider *R* as a function mapping an input string into a set of output strings, formally denoted by this type signature, where [Σ] equals *String*:

$$\mathcal{R} :: [\Sigma] \rightarrow \{[\Sigma]\} \qquad \mathcal{R} :: String \rightarrow \{String\} \qquad (1.1)$$

- A theoretical limitation of finite-state models of morphology is the problem of capturing **reduplication** of words or their elements (e.g., to express plurality) found in several human languages.
- Finite-state technology can be applied to the morphological modeling of isolating and agglutinative languages in a quite straightforward manner. Korean finite-state models are discussed by Kim, Lee and Rim, and Han, to mention a few.

Unification-Based Morphology

- The concepts and methods of these formalisms are often closely connected to those of logic programming.
- In finite-state morphological models, both surface and lexical forms are by themselves unstructured strings of atomic symbols.
- In higher-level approaches, linguistic information is expressed by more appropriate data structures that can include complex values or can be recursively nested if needed.
- Morphological parsing *P* thus associates linear forms φ with alternatives of structured content ψ, cf.

$$\mathcal{P} :: \phi \rightarrow \{\psi\} \qquad \mathcal{P} :: form \rightarrow \{content\} \qquad (1.2)$$

- Erjavec argues that for morphological modelling, word forms are best captured by regular expressions, while the linguistic content is best described through **typed feature structures**.
- Feature structures can be viewed as directed acyclic graphs.
- A node in a feature structure comprises a set of attributes whose values can be

- Nodes are associated with types, and atomic values are attributeless nodes distinguished by their type.
- Instead of unique instances of values everywhere, references can be used to establish value instance identity.
- Feature structures are usually displayed as attribute-value matrices or as nested symbolic expressions.
- Unification is the key operation by which feature structures can be merged into a more informative feature structure.
- Unification of feature structures can also fail, which means that the information in them is mutually incompatible.
- Morphological models of this kind are typically formulated as logic programs, and unification is used to solve the system of constraints imposed by the model.
- Advantages of this approach include better abstraction possibilities for developing a morphological grammar as well as elimination of redundant information from it.
- Unification-based models have been implemented for Russian, Czech, Slovene, Persian, Hebrew, Arabic, and other languages.

Functional Morphology

- Functional morphology defines its models using principles of functional programming and type theory.
- It treats morphological operations and processes as pure mathematical functions and organizes the linguistic as well as abstract elements of a model into distinct types of values and type classes.
- Though functional morphology is not limited to modelling particular types of morphologies in human languages, it is especially useful for fusional morphologies.
- Linguistic notions like paradigms, rules and exceptions, grammatical categories and parameters, lexemes, morphemes, and morphs can be represented intuitively(without conscious reasoning; instinctively) and succinctly(in a brief and clearly expressed manner) in this approach.
- Functional morphology implementations are intended to be reused as programming libraries capable of handling the complete morphology of a language and to be incorporated into various kinds of applications.

- Morphological parsing is just one usage of the system, the others being morphological generation, lexicon browsing, and so on.
- we can describe inflection *I*, derivation *D*, and lookup *L* as functions of these generic type

$$\mathcal{I} :: lexeme \rightarrow \{parameter\} \rightarrow \{form\} \quad (1.3)$$
$$\mathcal{D} :: lexeme \rightarrow \{parameter\} \rightarrow \{lexeme\} \quad (1.4)$$
$$\mathcal{L} :: content \rightarrow \{lexeme\} \quad (1.5)$$

- Many functional morphology implementations are embedded in a general-purpose programming language, which gives programmers more freedom with advanced programming techniques and allows them to develop full-featured, real-world applications for their models.
- The Zen toolkit for Sanskrit morphology is written in OCaml.
- It influenced the functional morphology framework in Haskell, with which morphologies of Latin, Swedish, Spanish, Urdu, and other languages have been implemented.
- In Haskell, in particular, developers can take advantage of its syntactic flexibility and design their own notation for the functional constructs that model the given problem.

- The notation then constitutes a so-called domain-specific embedded language, which makes programming even more fun.
- Even without the options provided by general-purpose programming languages, functional morphology models achieve high levels of abstraction.
- Morphological grammars in Grammatical Framework can be extended with descriptions of the syntax and semantics of a language.
- Grammatical Framework itself supports multilinguality, and models of more than a dozen languages are available in it as open-source software.

2.Finding structure of Documents

2.1 Introduction

- In human language, words and sentences do not appear randomly but have structure.
- For example, combinations of words from sentences- meaningful grammatical units, such as statements, requests, and commands.
- Automatic extraction of structure of documents helps subsequent NLP tasks: for example, parsing, machine translation, and semantic role labelling use sentences as the basic processing unit.
- Sentence boundary annotation(labelling) is also important for aiding human readability of automatic speech recognition (ASR) systems.
- Task of deciding where sentences start and end given a sequence of characters(made of words and typographical cues) **sentences boundary detection.**
- **Topic segmentation** as the task of determining when a topic starts and ends in a sequence of sentences.

- The statistical classification approaches that try to find the presence of sentence and topic boundaries given human-annotated training data, for segmentation.
- These methods base their predictions on features of the input: local characteristics that give evidence toward the presence or absence of a sentence, such as a period(.), a question mark(?), an exclamation mark(!), or another type of punctuation.
- Features are the core of classification approaches and require careful design and selection in order to be successful and prevent overfitting and noise problem.
- Most statistical approaches described here are language independent, every language is a challenging in itself.
- For example, for processing of Chinese documents, the processor may need to first segment the character sequences into words, as the words usually are not separated by a space.
- Similarly, for morphological rich languages, the word structure may need to be analyzed to extract additional features.
- Such processing is usually done in a pre-processing step, where a sequence of tokens is determined.
- Tokens can be word or sub-word units, depending on the task and language.
- These algorithms are then applied on tokens.

2.1.1 Sentence Boundary Detection

- **Sentence boundary detection** (Sentence segmentation) deals with automatically segmenting a sequence of word tokens into sentence units.
- In written text in English and some other languages, the beginning of a sentence is usually marked with an uppercase letter, and the end of a sentence is explicitly marked with a period(.), a question mark(?), an exclamation mark(!), or another type of punctuation.
- In addition to their role as sentence boundary markers, capitalized initial letters are used distinguish proper nouns, periods are used in abbreviations, and numbers and punctuation marks are used inside proper names.
- The period at the end of an abbreviation can mark a sentence boundary at the same time.
- Example: I spoke with Dr. Smith. and My house is on Mountain Dr.
- In the first sentence, the abbreviation Dr. does not end a sentence, and in the second it does.
- Especially **quoted sentences** are always problematic, as the speakers may have uttered multiple sentences, and sentence boundaries inside the quotes are also marked with punctuation marks.
- An automatic method that outputs word boundaries as ending sentences according to the presence of such punctuation marks would result in cutting some sentences incorrectly.

- Ambiguous abbreviations and capitalizations are not only problem of sentence segmentation in written text.
- Spontaneously written texts, such as short message service (SMS) texts or instant messaging(IM) texts, tend to be nongrammatical and have poorly used or missing punctuation, which makes sentence segmentation even more challenging.
- Similarly, if the text input to be segmented into sentences comes from an **automatic system**, such as optical character recognition (OCR) or ASR, that aims to translate images of handwritten, type written, or printed text or spoken utterances into machine editable text, the finding of sentences boundaries must deal with the errors of those systems as well.
- On the other hand, for conversational speech or text or multiparty meetings with ungrammatical sentences and disfluencies, in most cases it is not clear where the boundaries are.
- Code switching -that is, the use of words, phrases, or sentences from multiple languages by multilingual speakers- is another problem that can affect the characteristics of sentences.
- For example, when switching to a different language, the writer can either keep the punctuation rules from the first language or resort to the code of the second language.

- Conventional rule-based sentence segmentation systems in well-formed texts rely on patterns to identify potential ends of sentences and lists of abbreviations for disambiguating them.
- For example, if the word before the boundary is a known abbreviation, such as “Mr.” or “Gov.,” the text is not segmented at that position even though some periods are exceptions.
- To improve on such a rule-based approach, sentence segmentation is stated as a classification problem.
- Given the training data where all sentence boundaries are marked, we can train a classifier to recognize them.

2.1.2 Topic Boundary Detection

- **Segmentation**(Discourse or text segmentation) is the task of automatically dividing a stream of text or speech into topically homogenous blocks.
- This is, given a sequence of(written or spoken) words, the **aim of topic segmentation** is to find the boundaries where topics change.

- Topic segmentation is an important task for various language understanding applications, such as information extraction and retrieval and text summarization.
- For example, in information retrieval, if a long documents can be segmented into shorter, topically coherent segments, then only the segment that is about the user's query could be retrieved.
- During the late1990s, the U.S defence advanced research project agency(DARPA) initiated the **topic detection and tracking program** to further the state of the art in finding and following **new topic** in a stream of broadcast news stories.
- One of the tasks in the TDT effort was segmenting a **news stream into individual stories**.

2.2 Methods

- Sentence segmentation and topic segmentation have been considered as a **boundary classification problem**.
- Given a boundary candidate(between two word tokens for sentence segmentation and between two sentences for topic segmentation), the goal is to predict whether or not the candidate is an actual boundary (sentence or topic boundary).

- Formally, let **x∈X be the vector of features** (the observation) associated with a candidate and **y ∈Y be the label** predicted for that candidate.
- The label y can be **b for boundary** and $\bar{b}$ **for nonboundary**.
- Classification problem: given a set of training examples$(x,y)_{train}$, find a function that will assign the most accurate possible label y of unseen examples $x_{unseen.}$
- Alternatively to the binary classification problem, it is possible to model boundary types using finer-grained categories.
- For segmentation in text be framed as a three-class problem: sentence boundary b^a, without an abbreviation b^a and abbreviation not as a boundary $\bar{b}^a$.
- Similarly spoken language, a three way classification can be made between non-boundaries $\bar{b}$ statements b^s, and question boundaries b^q .
- For sentence or topic segmentation, the problem is defined as finding the most probable sentence or topic boundaries.
- The natural unit of sentence segmentation is words and of topic segmentation is sentence, as we can assume that topics typically do not change in the middle of a sentences.

- The words or sentences are then grouped into categories stretches belonging to one sentences or topic- that is word or sentence boundaries are classified into sentences or topic boundaries and -non-boundaries.
- The classification can be done at each potential boundary *i* (local modelling); then, the aim is to estimate the most probable boundary type $\hat{y}_i$ for each candidate x_i

$$\hat{y} = \underset{y_i \, in \, Y}{argmax} P(y_i|x_i)$$

Here, the ^ is used to denote estimated categories, and a variable without a ^ is used to show possible categories.

- In this formulation, a category is assigned to each example in isolation; hence, decision is made locally.
- However, the consecutive types can be related to each other. For example, in broadcast news speech, two consecutive sentences boundaries that form a single word sentence are very infrequent.
- In local modelling, features can be extracted from surrounding example context of the candidate boundary to model such dependencies.

- It is also possible to see the candidate boundaries as a sequence and search for the sequence of boundary types $\hat{Y} = \hat{y_1}, \ldots, \hat{y_n}$ that have the maximum probability given the candidate examples, $X = x_1, \ldots, x_n$:

$$\hat{Y} = \underset{y}{argmax} P(Y|X)$$

- We categorize the methods into local and sequence classification.
- Another categorization of methods is done according to the type of the machine learning algorithm: **generative versus discriminative**.
- Generative sequence models estimate the **joint distribution** of the observations P(X,Y) (words, punctuation) and the labels(sentence boundary, topic boundary).
- Discriminative sequence models, however, **focus on features** that categorize the differences between the labelling of that examples.

2.2.1 Generative Sequence Classification Methods

- Most commonly used generative sequence classification method for topic and sentence is the hidden Markov model (HMM).
- The probability in equation 2.2 is rewritten as the following, using the Bayes rule:

$$\hat{Y} = \underset{y}{argmax} P(Y|X) \quad 2.1$$

$$\hat{Y} = \underset{y}{argmax} P(Y|X) = \underset{y}{argmax} (P(X|Y)P(Y)/P(X)) = \underset{y}{argmax} (P(X|Y)P(Y) \quad 2.2$$

Here $\hat{Y}$= Predicted class(boundary) label

Y = (y_1,y_2,….y_k)= Set of class(boundary) labels

X = (x_1,x_2,….x_n)= set of feature vectors

P(Y|X) = the probability of given the X (feature vectors), what is the probability of X belongs to the class(boundary) label.

P(x) = Probability of word sequence

P(Y) = Probability of the class(boundary)

$$\hat{Y} = \underset{Y}{\operatorname{argmax}} P(Y|X) = \underset{Y}{\operatorname{argmax}} \frac{P(X|Y)P(Y)}{P(X)} = \underset{Y}{\operatorname{argmax}} P(X|Y)P(Y) \qquad (2.3)$$

- P(X) in the denominator is dropped because it is fixed for different Y and hence does not change the argument of max.
- P(X|Y) and P(Y) can be estimated as

$$P(X|Y) = \prod_{i=1}^{n} P(\mathbf{x}_i|y_1, \ldots, y_i) \tag{2.4}$$

and

$$P(Y) = \prod_{i=1}^{n} P(y_i|y_1, \ldots y_{i-1}) \tag{2.5}$$

2.2.2Discriminative Local Classification Methods

- Discriminative classifiers aim to model **$P(y_i | x_i)$ equation 2.1 directly**.
- The most important distinction is that whereas c**lass densities P(x|y)** are model assumptions **in generative approaches**, such as naïve Bayes, in discriminative methods, discriminant functions of the feature space define the model.
- A number of discriminative classification approaches, such as support vector machines, boosting, maximum entropy, and regression. Are based on very different machine learning algorithms.
- While discriminative approaches have been shown to outperform generative methods in many speech and language processing tasks.
- For **sentence segmentation, supervised learning methods have primarily been applied to newspaper articles**.
- Stamatatos, Fakotakis and Kokkinakis used **transformation based learning** (TBL) to infer rules for **finding sentence boundaries**.

- Many classifiers have been tried for the task: regression trees, neural networks, classification trees, maximum entropy classifiers, support vector machines, and naïve Bayes classifiers.
- The most Text tiling method Hearst for topic segmentation uses a l**exical cohesion metric** in a word vector space as an indicator of topic similarity.
- Figure depicts a typical graph of similarity with respect to **consecutive segmentation units**.

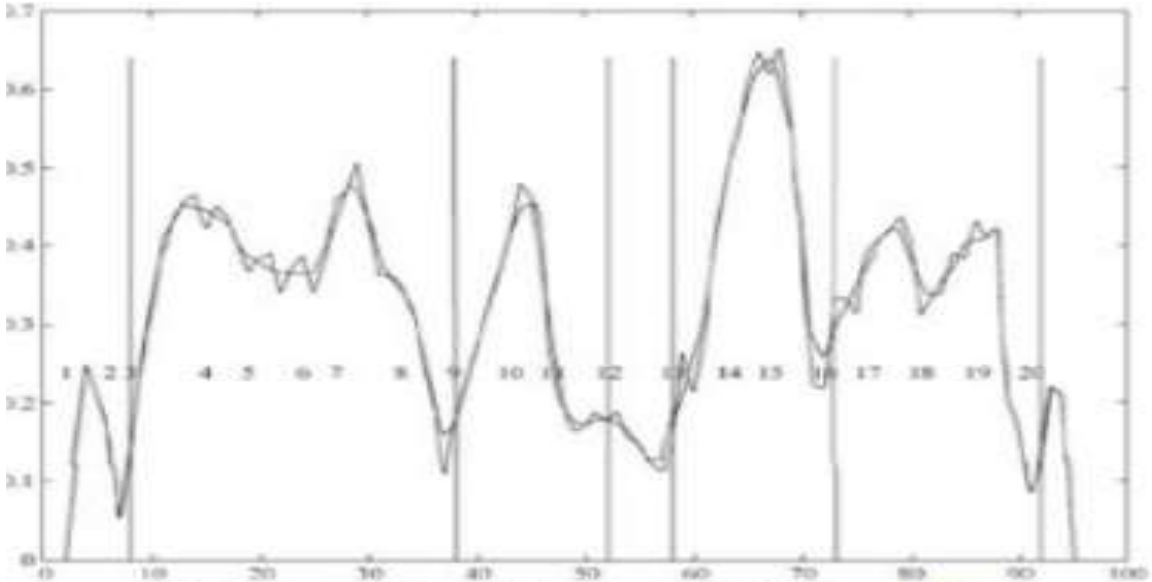


Figure 2–4. Text Tiling example (from [22])

Originally, two methods for computing the similarity scores were

- The document is chopped when the similarity is below some threshold.
- Originally, **two methods** for computing the similarity scores were proposed: **block comparison** and **vocabulary introduction**.

- The first, **block comparison, compares adjacent blocks of text to see how similar** they are according to how many words the adjacent blocks have in common.
- **Given two blocks, b_1 and b_2, each having k tokens** (sentences or paragraphs), the **similarity (or topical cohesion) score** is computed by the **formula**:

$$\frac{\sum_t w_{t,b_1} \cdot w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

- **Where $w_{t,b}$ is the weight assigned to term t in block b**.
- The weights can be binary or may be computed using other information retrieval- metrics such as term frequency.
- The second, the **vocabulary introduction method**, **assigns a score to a token-sequence gap** on the basis of **how many new words are seen in the interval in which it is the midpoint**.

- Similar to the **block comparison formulation, given two consecutive blocks b_1 and b_2, of equal number of words w**, the topical cohesion score is computed with the following formula:

$$\frac{NumNewTerms(b_1) + NumNewTerms(b_2)}{2 \times w}$$

- Where **NumNewTerms(b) returns the number of terms in block b seen the first time in text**.

2.2.3Discriminative Sequence Classification Methods

- In segmentation tasks, the sentence or topic decision for a given example(word, sentence, paragraph) highly depends on the decision for the examples in its vicinity.
- Discriminative sequence classification methods are in general extensions of local discriminative models with additional decoding stages that find the best assignment of labels by looking at neighbouring decisions to label.
- Conditional random fields(CRFs) are extension of maximum entropy, SVM struct is an extension of SVM, and maximum margin Markov networks(M^3N) are extensions of HMM.
- CRFs are a class of log-linear models for labelling structures.

- Contrary to local classifiers that predict sentences or topic boundaries independently, CRFs can oversee the whole sequence of boundary hypotheses to make their decisions.

Complexity of the Approaches

- The approaches described here have **advantages** and **disadvantages**.
- In a **given context** and under a set of observation features, one **approach may be better than other**.
- These approaches can be rated in terms of **complexity** (time and memory) of **their training** and **prediction algorithms** and in terms of their **performance on real-world datasets**.
- In terms of complexity, **training of discriminative approaches** is **more complex than training of generative ones** because they require multiple passes over the training data to adjust for feature weights.
- However, generative models such as HELMs can handle **multiple orders of magnitude larger training sets** and benefits, for instance, from decades of news wire transcripts.
- On the other hand, they work with **only a few features** (only words for HELM) and do not cope well with unseen events.

1.List and explain the challenges of morphological models. Mar 2021 [7]
2. Discuss the importance and goals of Natural Language Processing. Mar 2021 [8]
3.List the applications and challenges in NLP. Sep 2021 [7]
4.Explain any one Morphological model. Sep 2021 [8]
5.Discuss about challenging issues of Morphological model. Sep 2021 [7]
6.Differentiate between surface and deep structure in NLP with suitable examples. Sep 2021 [8]
7.Give some examples for early NLP systems. Sep 2021 [7]
8. Explain the performance of approaches in structure of documents? Sep 2021 [15]
9.With the help of a neat diagram, explain the representation of syntactic structure. Mar 2021 [8]
10.Elobarate the models for ambiguity resolution in Parsing. Mar 2021 [7]
11.Explain various types of parsers in NLP? Sep 2021 [8]
12.Discuss multilingual issues in detail. Sep 2021 [7]
13.Given the grammar S->AB|BB, A->CC|AB|a, B->BB|CA|b, C->BA|AA|b, word w='aabb'. Applay top down parsing test, word can be generated or not. Sep 2021 [8]
14.Explain Tree Banks and its role in parsing. Sep 2021 [7]

R18

Code No: 155CK

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

R18

Code No: 155CK

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B. Tech III Year I Semester Examinations, March - 2021

NATURAL LANGUAGE PROCESSING

(Computer Science and Engineering)

Time: 3 Hours

Max. Marks: 75

USED

List the applications in NLP.

Applications of NLP:

- Information retrieval & web search
- Grammar correction & Question answering
- Sentiment Analysis.
- Text Classification.
- Chatbots & Virtual Assistants.
- Text Extraction.
- Machine Translation.
- Text Summarization.
- Market Intelligence.
- Auto-Correct.

Discuss the importance and goals of Natural Language Processing.

Natural Language Processing
Unit-II

Syntax Analysis:

2.1Parsing Natural Language

2.2Treebanks: A Data-Driven Approach to Syntax,

2.3Representation of Syntactic Structure,

2.4Parsing Algorithms,

2.5Models for Ambiguity Resolution in Parsing, Multilingual Issues

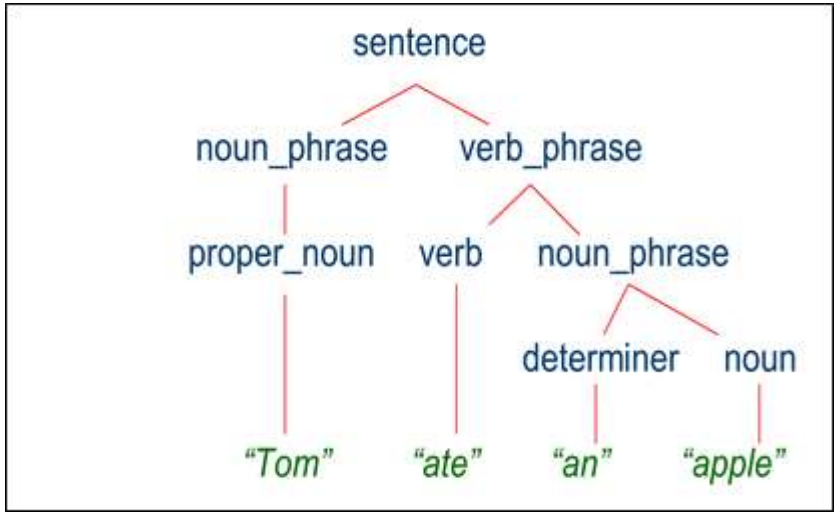
- The parsing in NLP is the process of determining the syntactic structure of a text by analysing its constituent words based on an underlying grammar.

Example Grammar:

```
sentence -> noun_phrase, verb_phrase
noun_phrase -> proper_noun
noun_phrase -> determiner, noun
verb_phrase -> verb, noun_phrase
proper_noun -> [Tom]
noun -> [apple]
verb -> [ate]
determiner -> [an]
```

- Then, the **outcome** of the parsing process would be a **parse tree**, where **sentence** is the root, intermediate nodes such as **noun_phrase**, **verb_phrase** etc. have children - hence they are called **non-terminals** and finally, the leaves of the tree **'Tom'**, '**ate'**, '**an'**, '**apple'** are called **terminals**.

Parse Tree:



- A treebank can be defined as a linguistically annotated corpus that includes some kind of syntactic analysis over and above part-of-speech tagging.
- A sentence is parsed by relating each word to other words in the sentence which depend on it.
- The syntactic parsing of a sentence consists of finding the correct syntactic structure of that sentence in the given formalism/grammar.
- Dependency grammar (DG) and phrase structure grammar(PSG) are two such formalisms.
- PSG breaks sentence into constituents (phrases), which are then broken into smaller constituents.
- Describe phrase, clause structure Example: NP,PP,VP etc.,
- DG: syntactic structure consist of lexical items, linked by binary asymmetric relations called dependencies.
- Interested in grammatical relations between individual words.
- Does propose a recursive structure rather a network of relations
- These relations can also have labels.

Constituency tree vs Dependency tree

- Dependency structures explicitly represent
- Head-dependent relations (directed arcs)
- Functional categories (arc labels)
- Possibly some structural categories (POS)
- Phrase structure explicitly represent
- Phrases (non-terminal nodes)
- Structural categories (non-terminal labels)
- Possible some functional categories (grammatical functions)

Defining candidate dependency trees for an input sentence

♦ Learning: scoring possible dependency graphs for a given sentence, usually by factoring the graphs into their component arcs

♦ Parsing: searching for the highest scoring graph for a given sentence

Syntax

- In NLP, the syntactic analysis of natural language input can vary from being very low-level, such as simply **tagging** each word in the sentence with a **part of speech** (POS), or very high level, such as **full parsing**.
- In **syntactic parsing, ambiguity** is a particularly difficult problem because the most possible analysis has to be chosen from an exponentially large number of alternative analyses.
- From **tagging** to **full parsing**, algorithms that can handle such ambiguity have to be carefully chosen.
- Here we explores the syntactic analysis methods from tagging to full parsing and the use of supervised machine learning to deal with ambiguity.

2.1Parsing Natural Language

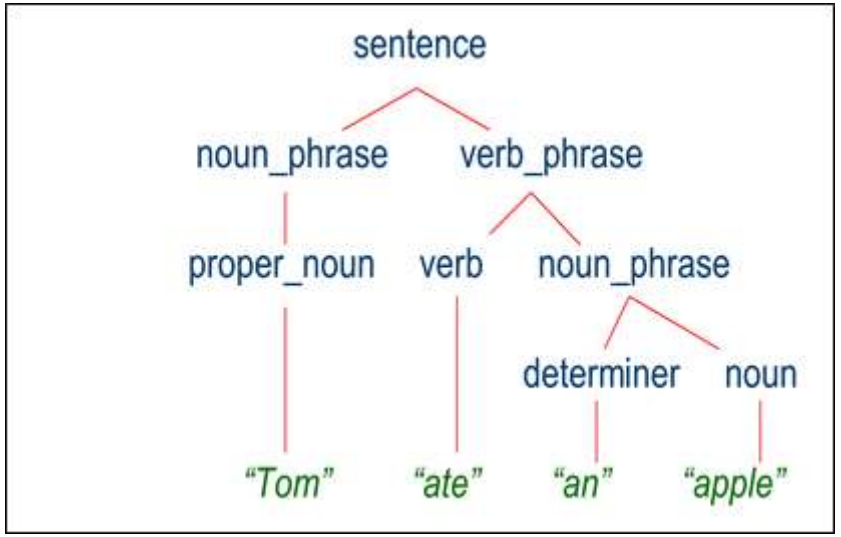
- In a **text-to-speech** application, input sentences are to be converted to a spoken output that should sound like it was spoken by a native speaker of the language.
- Example: **He wanted to go a drive in the country**.
- There is a natural pause between the words **derive** and **In** in sentence that reflects an underlying hidden structure to the sentence.
- **Parsing** can provide a structural description that identifies such a break in the intonation.

- A simpler case: The cat who **lives** dangerously had nine **lives**.
- In this case, a text-to-speech system needs to know that the first instance of the word lives is a **verb** and the second instance is a **noun** before it can begin to produce the natural intonation for this sentence.
- This is **an instance** of the **part-of-speech (POS) tagging problem** where each word in the sentence is assigned a most likely part of speech.
- Another motivation for parsing comes from the natural language **task of summarization**, in which **several documents about the same topic** should be condensed down to a small digest of information.
- Such a summary may be in response to a question that is answered in the set of documents.
- In this case, a useful **subtask** is to **compress an individual sentence** so that only the relevant portions of a sentence is included in the summary.
- For example:

  Beyond the basic level, the **operations of the** three **products vary** widely.

  The operations of the products vary.
- The elegant way to approach this task is to first **parse the sentence** to find the **various constituents**: where we recursively partition the words in the sentence into individual phrases such as a verb phrase or a noun phrase.

- The output of the parser for the input sentence is shown in Fig.



- Another example is the paragraph parsing.
- In the sentence fragment, the **capitalized phrase** EUROPEAN COUNTRIES can be **replaced** with other phrases **without changing** the essential meaning of the sentences.
- A few examples of replacement phrases are shown in the sentence fragments .

  Open border imply increasing racial fragmentation in EUROPEAN COUNTRIES.
-

Open borders imply increasing racial fragmentation in the countries of Europe
Open borders imply increasing racial fragmentation in European states.
Open borders imply increasing racial fragmentation in Europe
Open borders imply increasing racial fragmentation in European nations
Open borders imply increasing racial fragmentation in European countries.

- In contemporary NLP, syntactic parsers are routinely used in many applications, including but not limited to statistical machine translation, **information extraction** from text collections, **language summarizations**, producing entity grinds for language generation, **error correction** in text.
-

2.2Treebanks: A Data-Driven Approach to Syntax

- Parsing recovers information that is not explicit in the input sentence.
- This implies that a parser requires some **knowledge(syntactic rules)** in addition to the input sentence about the kind of syntactic analysis that should be produced as output.
- **One method to provide such knowledge** to the parser is to write down a grammar of the language – **a set of rules** of syntactic analysis **as a CFGs**.
- In natural language, it is far **too complex** to simply **list all the syntactic rules** in terms of a CFG.
- The second knowledge acquisition problem- **not only do we need to know** the **syntactic rules** for a particular language, but we also need to know which **analysis is the most plausible**(probably) for a given input sentence.
- The **construction of treebank is a data driven approach to syntax analysis** that allows us to address both of these knowledge acquisition bottlenecks in one stroke.
- A treebank is simply a **collection of sentences** (also called a corpus of text), where each sentence is provided a **complete syntax analysis**.
- The syntactic analysis for each sentence has been judged by a **human expert** as the most possible analysis for that sentence.

- A lot of care is taken during the **human annotation process** to ensure that a consistent treatment is provided across the treebank for related grammatical phenomena.
- There is no **set of syntactic rules** or **linguistic grammar** explicitly provided by a treebank, and typically there is no list of syntactic constructions provided explicitly in a treebank.
- **A detailed set of assumptions about the syntax is typically used** as an **annotation** guideline to help the human experts produce the **single-most plausible syntactic analysis** for each sentence in the corpus.
- Treebanks provide a **solution to the two kinds** of knowledge acquisition **bottlenecks**.
- Treebanks solve the first knowledge acquisition problem of **finding the grammar** underlying the syntax analysis because the **syntactic analysis is directly** given instead of a grammar.
- In fact, the parser does not necessarily need any explicit grammar rules as long as it can faithfully produce a syntax analysis for an input sentence.
- Treebank solve the second knowledge acquisition problem as well.
- Because each sentence in a treebank has been given its **most plausible(probable)** syntactic analysis, **supervised machine learning methods** can be used **to learn a scoring function** over **all possible** syntax analyses.

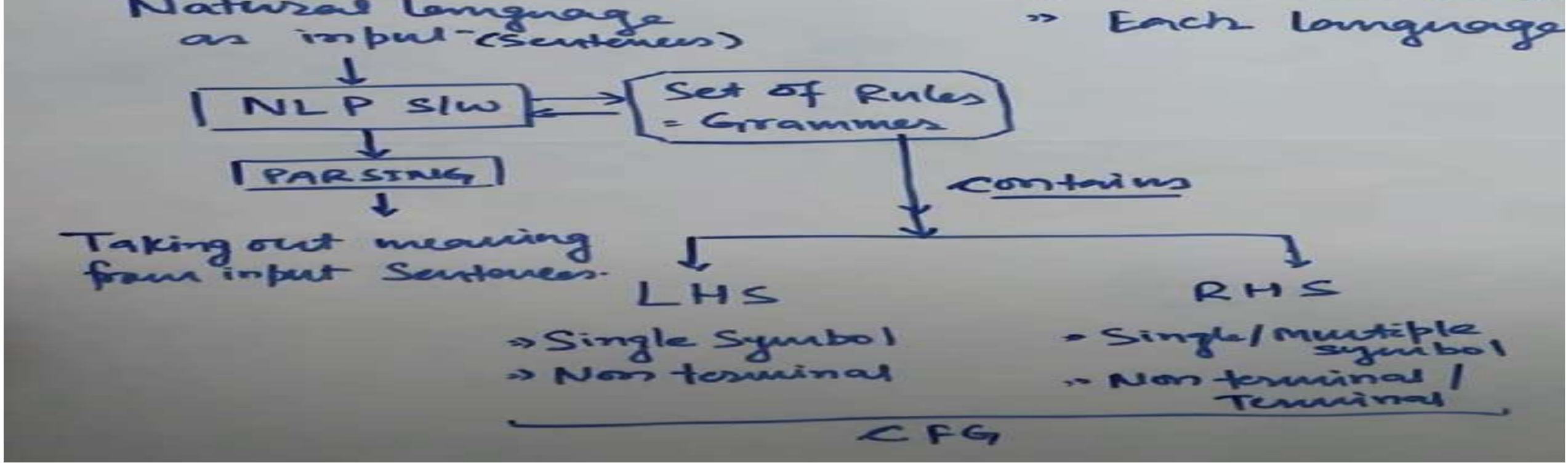
- **Two main approaches** to syntax analysis are used to construct treebanks: **dependency graph** and **phrase structure trees**.
- These two representations are **very closely** related to each other and under some assumptions, one representation can be converted to another.
- **Dependence analysis** is typically **favoured** for languages such as **Czech** and **Turkish**, that have free word order.
- **Phrase structure analysis** is often used to provide additional information about **long-distance dependencies** and mostly languages like **English** and **French**.

- NLP: is the capability of the computer software to understand the natural language.
- There are variety of languages in the world.
- Each language has its own structure(SVO or SOV)->called grammar ->has certain set of rules->determines: what is allowed, what is not allowed.
- English: S O V Other languages: S V O or O S V
  I eat mango
- Grammar is defined as the rules for forming well-structured sentences.
- belongs to V_N
- Different Types of Grammar in NLP

1.Context-Free Grammar (CFG) 2.Constituency Grammar (CG) or Phrase structure grammar 3.Dependency Grammar (DG)

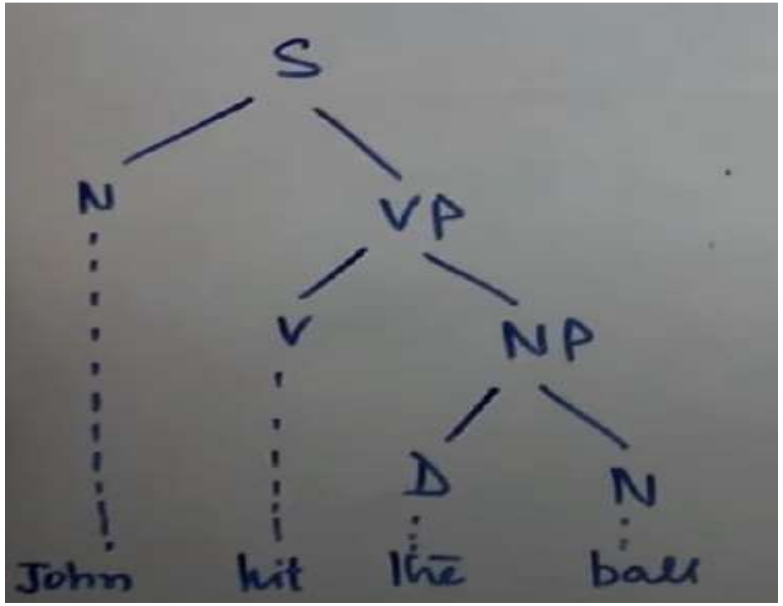
Context-Free Grammar (CFG)

- Mathematically, a grammar G can be written as a 4-tuple (N, T, S, P)
- **N or V_N =** set of non-terminal symbols, or variables.
- **T or $\sum$ =** set of terminal symbols.
- **S =** Start symbol where $S \in N$
- **P =** Production rules for Terminals as well as Non-terminals.
- It has the form $\alpha \rightarrow \beta$, where α and β are strings on $V_N \cup \sum$ at least one symbol of α

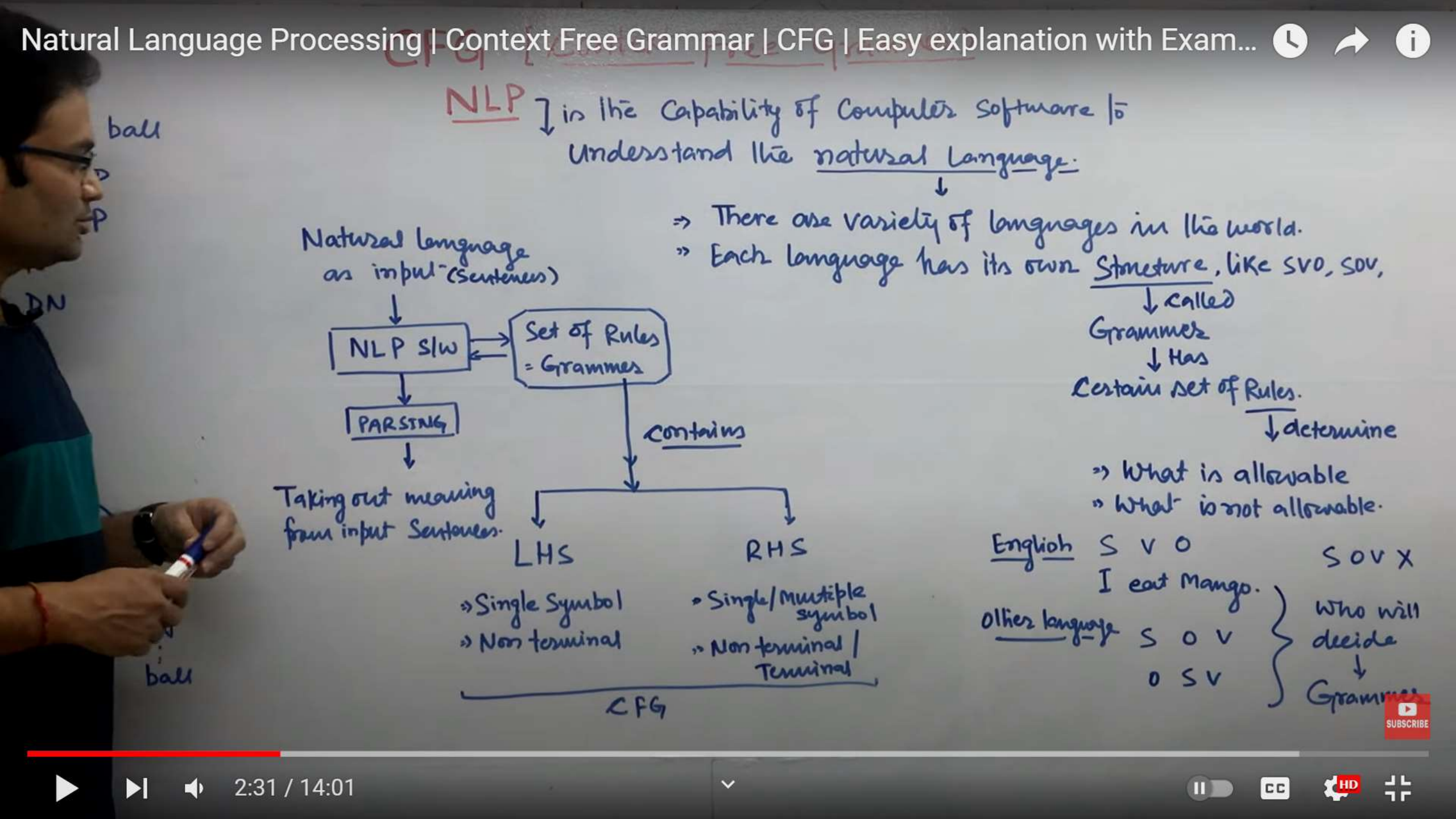


- Example: Jogn hit the ball

S -> NP VP
VP -> V NP
V ->hit
NP-> DN
D->the
N->John|ball



Natural Language Processing | Context Free Grammar | CFG | Easy explanation with Exam...



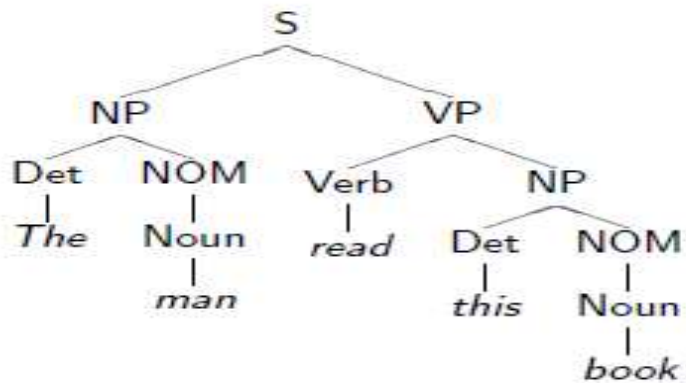
2:31 / 14:01

Application of grammar rewrite rules

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

S → NP VP
→ Det NOM VP
→ *The* NOM VP
→ *The* Noun VP
→ *The man* VP
→ *The man* Verb NP
→ *The man read* NP
→ *The man read* Det NOM
→ *The man read this* NOM
→ *The man read this* Noun
→ *The man read this book*

Parse tree



2.3 Representation of Syntactic Structure

2.3.1 Syntax Analysis Using Dependency Graphs

- The main philosophy behind dependency graphs is **to connect a word- the head of a phrase**- with the dependents in that phrase.
- The notation connects a head with its dependent using a directed (asymmetric) connections.
- Dependency graphs, just like phrase structures trees, is a representation that is consistent with many different linguistic frameworks.

- The **words** in the input sentence are treated as the only **vertices** in the graph, which are linked together by **directed arcs** representing syntactic **dependencies**.
- In dependency-based syntactic parsing, the task is to derive a syntactic structure for an input sentence by **identifying** the **syntactic head of the each word** in the sentence.
- This defines a dependency graph, where the **nodes are the words** of the input sentence and **arcs** are the **binary relations** from **head to dependent**.

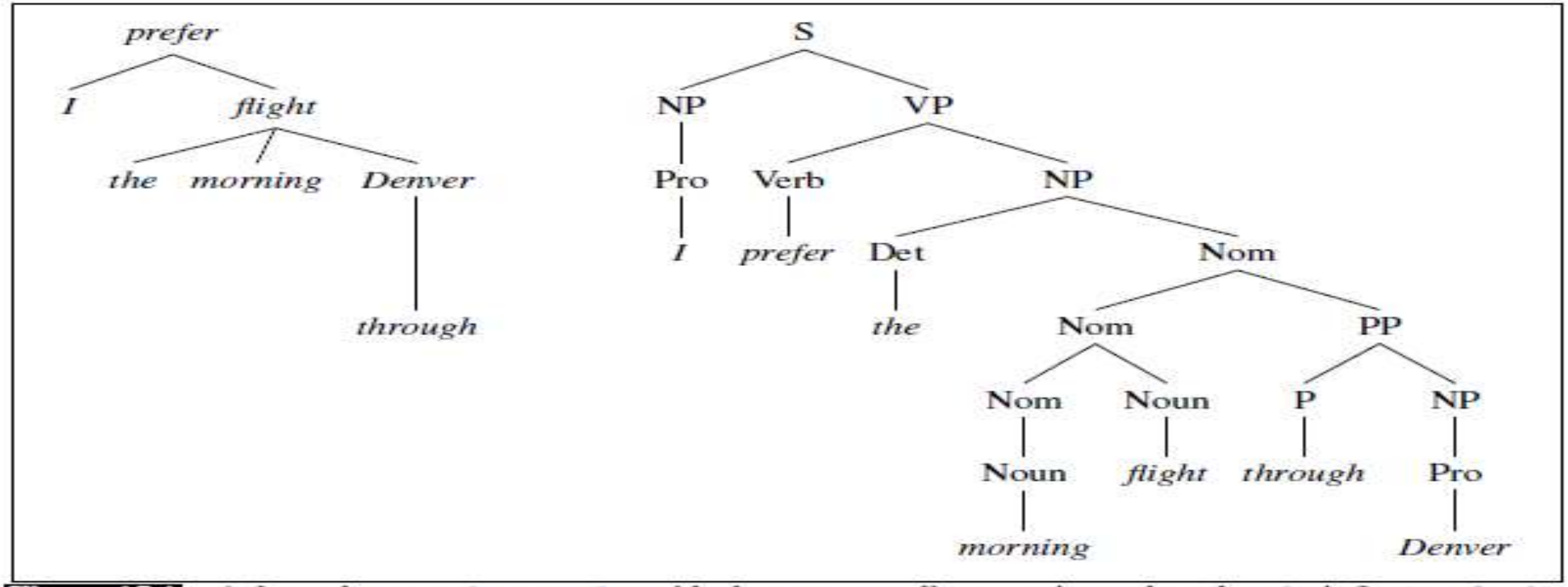
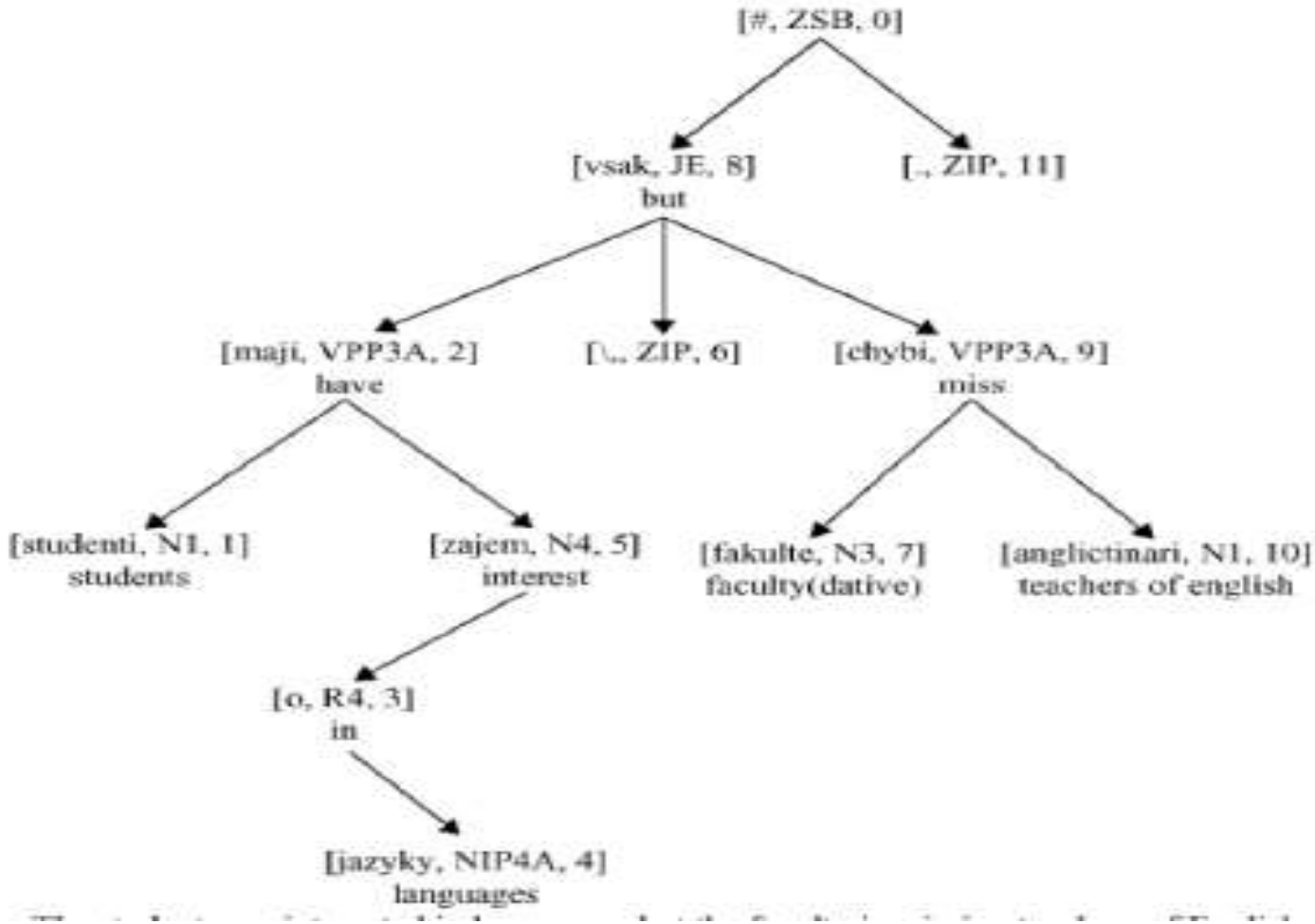


Figure 15.1 A dependency-style parse alongside the corresponding constituent-based analysis for *I prefer the morning flight through Denver.*

- The dependency tree analyses, where **each word depends on exactly one parent**, either another word or a dummy root symbol.
- By convention, in dependency tree **0 index** is used to indicate the **root symbol** and the directed arcs are drawn from the head word to the dependent word.
- In the Fig shows a dependency tree for Czech sentence taken from the Prague dependency treebank.



- Each node in the graph is a word, its part of speech and the position of the word in the sentence.
- For example [fakulte, N3,7] is the seventh word in the sentence with POS tag N3.
- The node [#, ZSB,0] is the root node of the dependency tree.
- There are many **variations** of dependency syntactic analysis, but the **basic textual format** for a dependency tree can be written in the following form.
- Where each dependent **word specifies** the **head Word** in the sentence, and **exactly one word** is dependent to the root of the sentence.

Index	Word	Part of Speech	Head	Label
1	They	PRP	2	SBJ
2	persuaded	VBD	0	ROOT
3	Mr.	NNP	4	NMOD
4	Trotter	NNP	2	IOBJ
5	to	TO	6	VMOD
6	take	VB	2	OBJ
7	it	PRP	6	OBJ
8	back	RB	6	PRT
9	.	.	2	P

- An important **notation** in dependency analysis is the notation of **projectivity**, which is a **constraint** imposed by the **linear order** of words on the dependencies between words.
- A projective dependency tree is one where if we put the words in a **linear order** based on the sentence with the **root symbol** in the **first position**, the dependency arcs can be drawn above the words without any crossing dependencies.

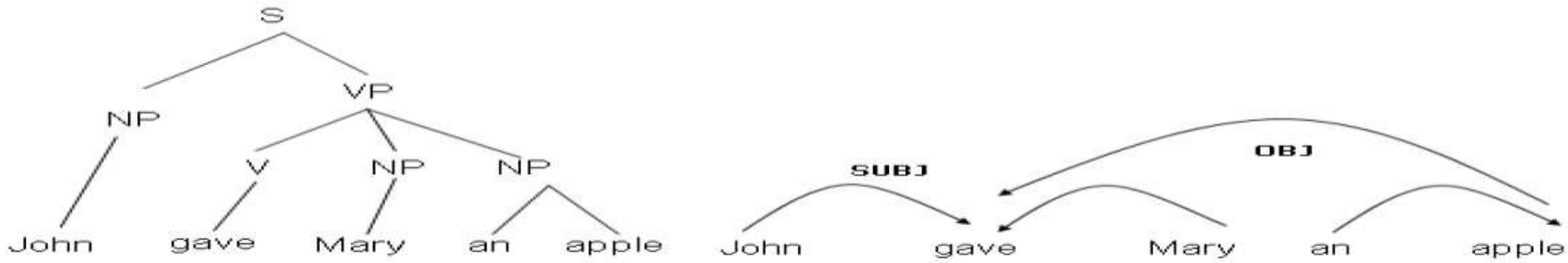


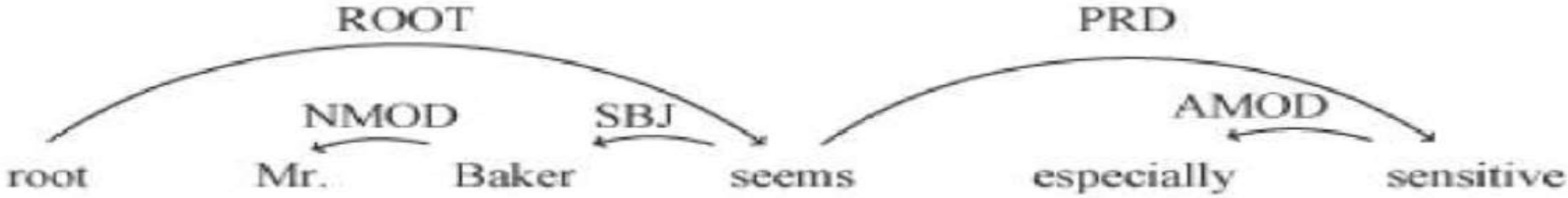
Figure 1
Phrase structure tree and the corresponding Dependency structure tree

2.3.2 Syntax Analysis Using Phrase Structures Trees

- A Phrase Structure syntax analysis of a sentence derives from the traditional sentence diagrams that partition a sentence into constituents, and larger constituents are formed by meaning smaller ones.
- Phrase structure analysis also typically incorporate ideas from generative grammar(from linguistics) to deal with displaced constituents or apparent long-distance relationships between heads and constituents.
- A phrase structure tree can be viewed as implicitly having a predicate-argument structure associated with it.
- Sentence includes a subject and a predicate. The subject is a noun phrase (NP) and the predicate is a verb phrase.
- For example, the phrase structure analysis : ***Mr. Baker seems especially sensitive***, taken from the Penn Treebank.
- The subject of the sentence is marked with the SBJ marker and predicate of the sentence is marked with the PRD marker.
-

```
(S (NP-SBJ (NNP Mr.)
           (NNP Baker))
   (VP (VBZ seems)
       (ADJP-PRD (RB especially)
                 (JJ sensitive))))
```

- NNP: proper noun, singular VBZ: verb, third person singular present
  ADJP: adjective phrase RB: adverb JJ:adjective
- The same sentence gets the following dependency tree analysis: some of the information from the bracketing labels from the phrase structure analysis gets mapped onto the labelled arcs of the dependency analysis.



- To explain some details of phrase structure analysis in treebank, which was a project that annotated 40,000 sentences from the wall street journal with phrase structure tree,

```
(SBARQ (WHNP-1 What)
       (SQ is (NP-SBJ Tim)
              (VP eating (NP *T*-1)))
       ?)
```

- The SBARQ label marks what questions ie those that contain a gap and therefore require a trace.

- Wh- moved noun phrases are labeled WHNP and put inside SBARQ. They bear an identity index that matches the reference index on the *T* in the position of the gap.
- However questions that are missing both subject and auxiliary are label SQ
- NP-SBJ noun phrases cab be subjects.
- *T* traces for wh- movement and this empty trace has an index (here it is 1) and associated with the WHNP constituent with the same index.

Parsing Algorithms

- Given an input sentence, a parser produces an output analysis of that sentence.
- Treebank parsers do not need to have an explicit grammar, but to discuss the parsing algorithms simpler, we use CFG.
- The simple CFG G that can be used to derive string such as a and b or c from the start symbol N.

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

- An important concept for parsing is a derivation.
- For the input string a and b or c, the following sequence of actions separated by `=>` symbol represents a sequence of steps called derivation.

```
N
=> N 'or' N
=> N 'or c'
=> N 'and' N 'or c'
=> N 'and b or c'
=> 'a and b or c'
```

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

- In this derivation, each line is called a sentential form.
- In the above derivation, we restricted ourselves to only expanded on the rightmost nonterminal in each sentential form.
- This method is called the rightmost derivation of the input using a CFG.
- This derivation sequence exactly corresponds to the construction of the following parse tree from left to right, one symbol at a time.

```
(N (N (N a)
      and
      (N b))
   or
   (N c))
```

- However, a unique derivation sequence is not guaranteed.
- There can be many different derivation.
- For example, one more rightmost derivation that results following parse tree.

```
(N (N a)
   and
   (N (N b)
      or
      (N c)))

'a and b or c'
=> N 'and b or c'        # use rule N -> a
=> N 'and' N 'or c'      # use rule N -> b
=> N 'and' N 'or' N      # use rule N -> c
=> N 'and' N             # use rule N -> N or N
=> N                     # use rule N -> N and N
```

Shift Reduce Parsing

- To build a parser, we need an algorithm that can perform the steps in the above rightmost derivation for any grammar and for any input string.
- Every CFG turns out to have an automata that is equivalent to it, called pushdown automata (just like regular expression can be converted to finite-state automata).
- An algorithm for parsing that is general for any given CFG and input string.
- The algorithm is called shift-reduce parsing which uses two data structures: a buffer for input symbols and a stack for storing CFG symbols.

Shift Reduce Parser

Start with the sentence to be parsed in an input buffer.

- a "shift" action correponds to pushing the next input symbol from the buffer onto the stack
- a "reduce" action occurrs when we have a rule's RHS on top of the stack. To perform the reduction, we pop the rule's RHS off the stack and replace it with the terminal on the LHS of the corresponding rule.

(When either "shift" or "reduce" is possible, choose one arbitrarily.)

If you end up with only the START symbol on the stack, then success!
If you don't, and you cannot and no "shift" or "reduce" actions are possible, backtrack.

		a and b or c	Init
a	a	and b or c	shift a
(N a)	N	and b or c	reduce N -> a
(N a) and	N and	b or c	shift and
(N a) and b	N and b	or c	shift b
(N a) and (N b)	N and N	or c	reduce N -> b
(N (N a) and (N b))	N	or c	reduce N -> a
(N (N a) and (N b)) or	N or	c	shift or
(N (N a) and (N b)) or c	N or c		shift c
(N (N a) and (N b)) or (N c)	N or N		reduce N -> c
(N (N (N a) and (N b)) or (N c))	N		reduce N -> N or N
(N (N (N a) and (N b)) or (N c))	N		Accept!

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

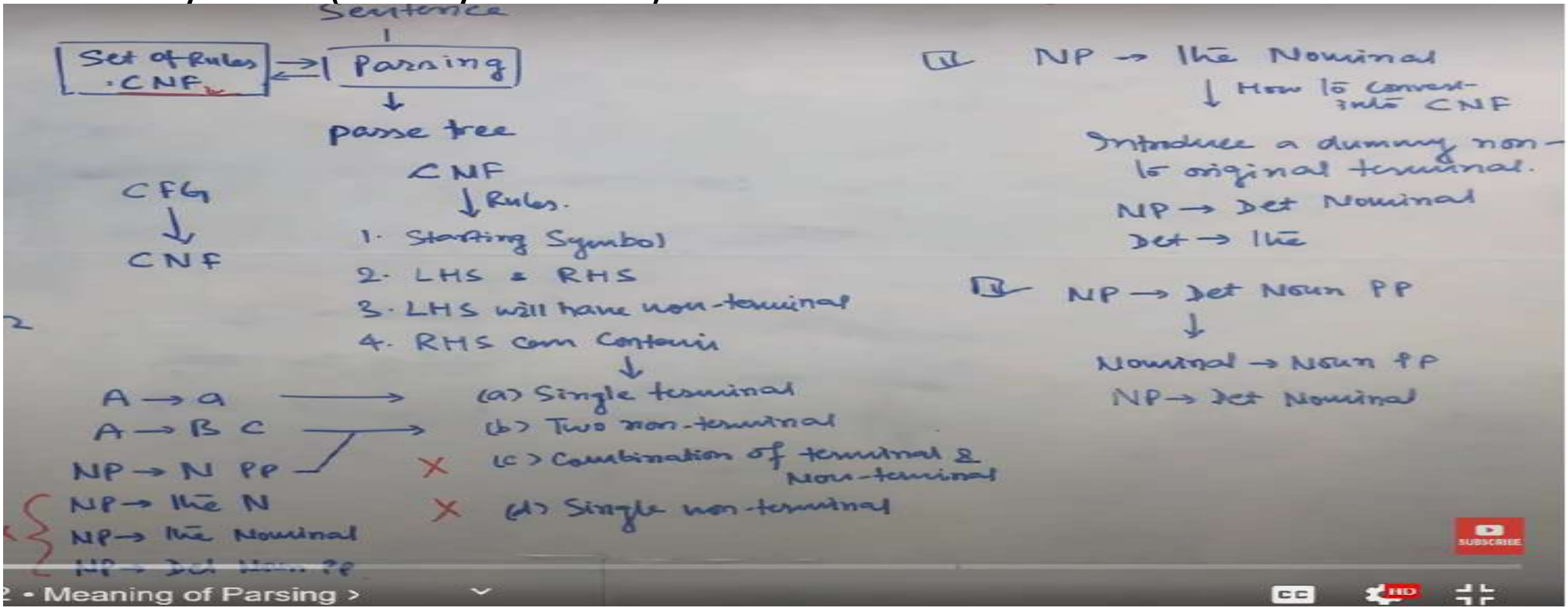
S.No	Parse Tree	Stack	Input	Action
1			a and b or c	Init
2	a	a	and b or c	Shift a
3	(N a)	N	and b or c	Reduce N->a
4	(N a) and	N and	b or c	Shift and
	(N a) and b	N and b	or c	Shift b
	(N a) and (N b)	N and N	or c	Reduce N->b
	(N (N a) and (N b))	N	or c	Reduce N->N and N
	(N (N a) and (N b)) or	N or	c	Shift or
9	(N (N a) and (N b)) or c	N or c		Shift c
10	(N (N a) and (N b)) or (N c)	N or N		Reduce N->c
11	(N (N (N a) and (N b)) or (N c))	N		Reduce N->N or N

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

1. Start with an empty stack and the buffer contains the input string.
2. Exit with success if the top of the stack contains the start symbol of the grammar and if the buffer is empty.
3. Choose between the following two steps (if the choice is ambiguous, choose one based on an oracle):
   - Shift a symbol from the buffer onto the stack.
   - If the top k symbols of the stack are $\alpha_1 \ldots \alpha_k$ which corresponds to the right-hand side of a CFG rule $A \rightarrow \alpha_1 \ldots \alpha_k$ then replace the top k symbols with the left-hand side non-terminal A.
4. Exit with failure if no action can be taken in previous step.
5. Else, go to Step 2.

Hypergraphs and Chart Parsing(CYK Parsing)

- CFG s in the worst case such a parser might have to resort to backtracking, which means re-parsing the input which leads to a time that is exponential in the grammar size in the worst case.
- Variants of this algorithm(CYK) are often used in statistical parsers that attempt to search the space of possible parse trees without the limitation of purely left to right parsing.
- One of the earliest recognition parsing algorithm is CYK (Cocke, Kasami and younger) parsing algorithm and It works only with CNF(Chomsky normal form).



Sentence

↓

Set of Rules · CNF ⇄ Parsing

↓

passe tree

CFG

↓

CNF

CNF

↓ Rules.

1. Starting Symbol
2. LHS & RHS
3. LHS will have non-terminal
4. RHS can Contain

↓

(a) Single terminal

(b) Two non-terminal

(c) Combination of terminal & Non-terminal

(d) Single non-terminal

A → a ⟶ (a)

A → B C ⟶ (b)

NP → N PP ✗ (c)

NP → Ika N ✗ (d)

NP → Ika Nominal

NP → Det Noun PP

[II] NP → Ika Nominal

↓ How to convert into CNF

Introduce a dummy non-t⁰ original terminal.

NP → Det Nominal

Det → Ika

[III] NP → Det Noun PP

↓

Nominal → Noun PP

NP → Det Nominal

SUBSCRIBE

• Meaning of Parsing >

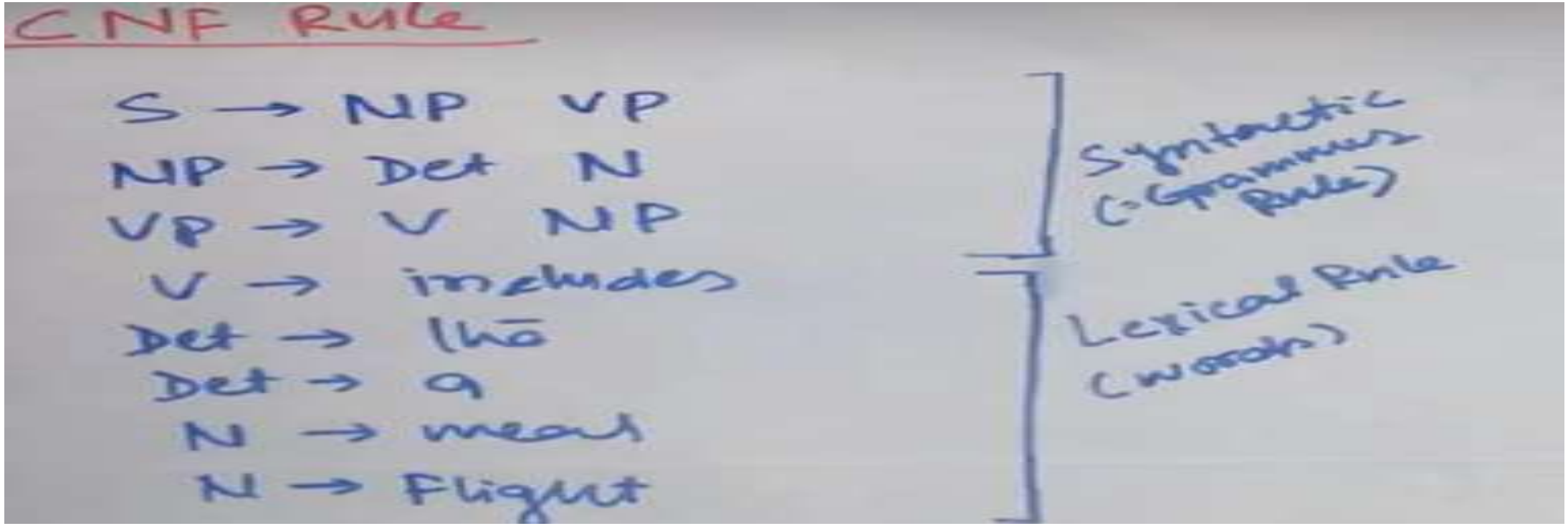
Our example CFG G:

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

is re-written into a new CFG G_c where the right hand side only contains up to two non-terminals. This is done by introducing two new non-terminals N^ and Nv:

```
N -> N N^
N^ -> 'and' N
N -> N Nv
Nv -> 'or' N
N -> 'a' | 'b' | 'c'
```

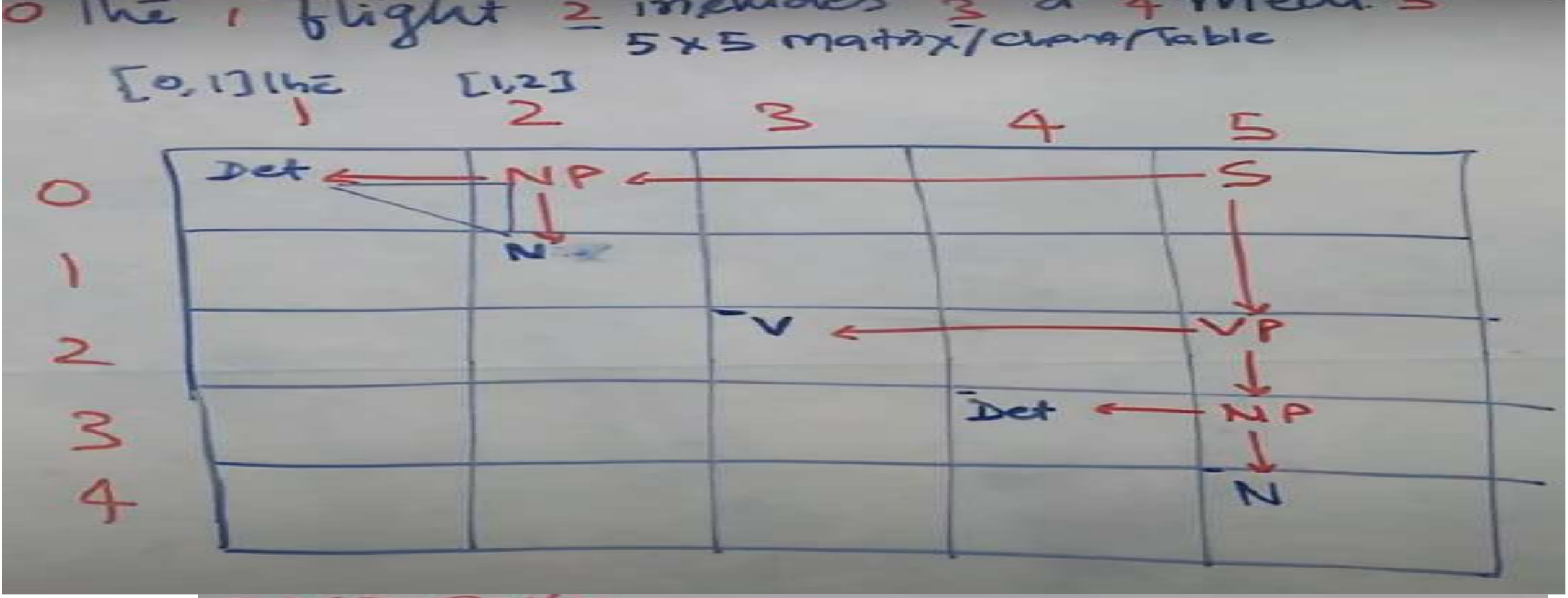
CYK example:



0 the 1 flight 2 includes 3 a 4 meal 5

5 x 5 matrix / chart / table

[0,1] the [1,2]



CNF RULE

S → NP VP
NP → Det N
VP → V NP
V → includes
Det → the
Det → a
N → meal
N → flight

Syntactic (Grammar Rule)

Lexical Rule (words)

G_f that represents the *forest* of parse trees is shown below. Imagine that the input string is broken up into spans *0 a 1 and 2 b 3 or 4 c 5* so that *a* is span 0,1 and the string *b or c* is the span 2,5 in this string. The non-terminals in this forest grammar G_c include the span information. The different parse trees that can be generated using this grammar are the valid parse trees for the input sentence.

```
N[0,5] -> N[0,1] N^[1,5]
N[0,3] -> N[0,1] N^[1,3]
N^[1,3] -> 'and'[1,2] N[2,3]
N^[1,5] -> 'and'[1,2] N[2,5]
N[0,5] -> N[0,3] Nv[3,5]
N[2,5] -> N[2,3] Nv[3,5]
Nv[3,5] -> 'or'[3,4] N[4,5]
N[0,1] -> 'a'[0,1]
N[2,3] -> 'b'[2,3]
N[4,5] -> 'c'[4,5]
```

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

is re-written into a new CFG G_c where the right hand side only contains up to two non-terminals. This is done by introducing two new non-terminals `N^` and `Nv`:

```
N -> N N^
N^ -> 'and' N
N -> N Nv
Nv -> 'or' N
N -> 'a' | 'b' | 'c'
```

In this view, a parsing algorithm is defined as taking as input a CFG and an input string and producing a specialized CFG that is a compact representation of all legal parses for the input. A parser has to create all the valid specialized rules or alternatively create a path from the start symbol non-terminal that spans the entire string to the leaf nodes that are the input tokens.

Let us examine the steps the parser has to take to construct a specialized CFG. First let us consider the rules that generate only lexical items:

```
N[0,1] -> 'a'[0,1]
N[2,3] -> 'b'[2,3]
N[4,5] -> 'c'[4,5]
```

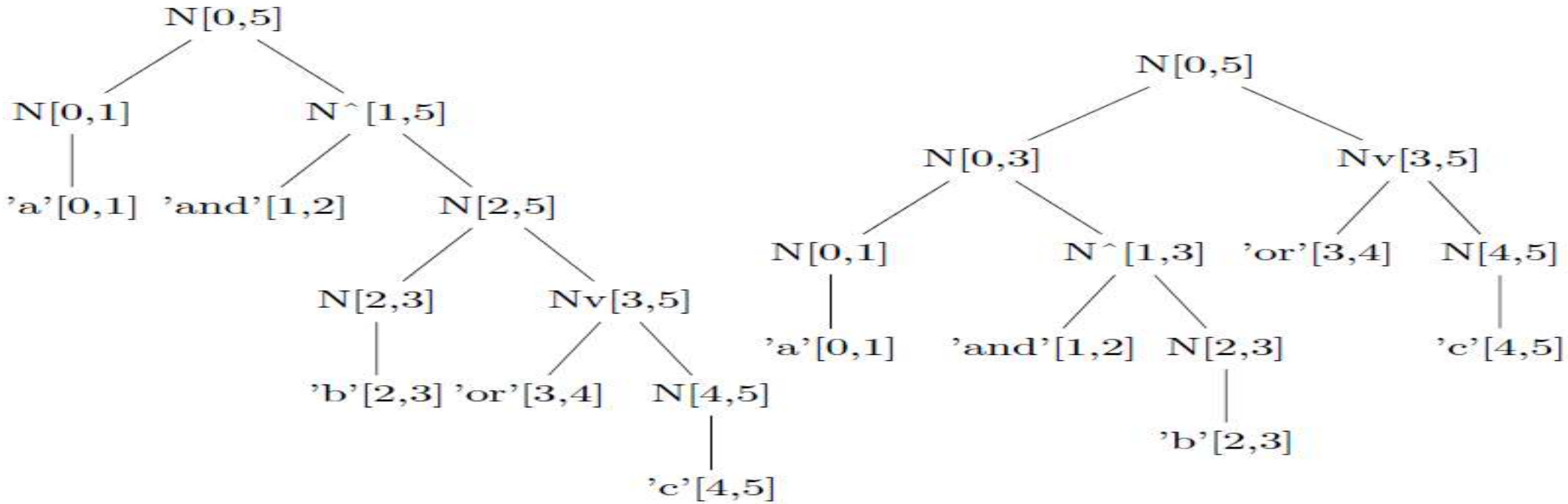


Figure 1.8. Parse trees embedded in the *specialized* CFG for a particular input string. The nodes with the same label, e.g. *N[0,5]* can be merged to form a hypergraph representation of all parses for the input.

```
N[0,5] -> N[0,1] N^[1,5]
N[0,3] -> N[0,1] N^[1,3]
N^[1,3] -> 'and'[1,2] N[2,3]
N^[1,5] -> 'and'[1,2] N[2,5]
N[0,5] -> N[0,3] Nv[3,5]
N[2,5] -> N[2,3] Nv[3,5]
Nv[3,5] -> 'or'[3,4] N[4,5]
N[0,1] -> 'a'[0,1]
N[2,3] -> 'b'[2,3]
N[4,5] -> 'c'[4,5]
```

```
N -> N 'and' N
N -> N 'or' N
N -> 'a' | 'b' | 'c'
```

is re-written into a new CFG G_c where the right hand side only contains up to two non-terminals. This is done by introducing two new non-terminals `N^` and `Nv`:

```
N -> N N^
N^ -> 'and' N
N -> N Nv
Nv -> 'or' N
N -> 'a' | 'b' | 'c'
```

CNF Rule

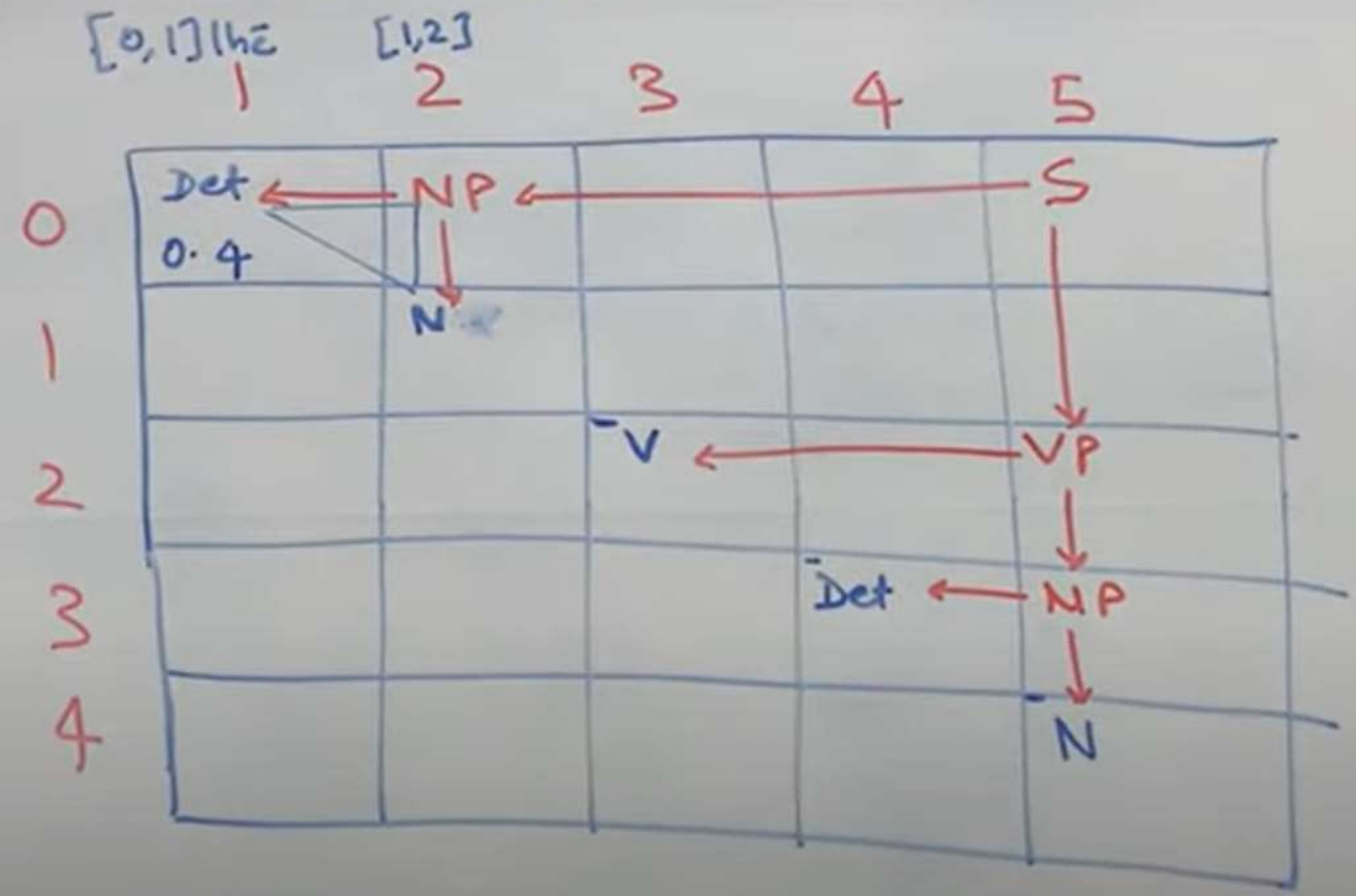
S → NP VP [0.80]
NP → Det N [0.3]
VP → V NP [0.20]
V → includes [0.05]
Det → lhō [0.4]
Det → a [0.4]
N → meal [0.01]
N → flight [0.02]

Syntactic (:Grammar Rule)

Lexical Rule (words)

0 lhē 1 flight 2 includes 3 a 4 meal. 5

5x5 matrix/chart/table



SUBSCRIBE

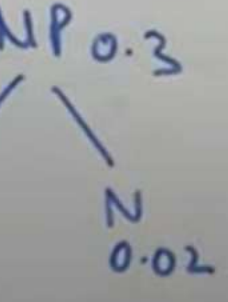
CNF Rule

S → NP VP [0.80]
NP → Det N [0.3]
VP → V NP [0.20]
V → includes [0.05]
Det → lhā [0.4]
Det → q [0.4]
N → meal [0.01]
N → Flight [0.02]

Syntactic (=Grammar Rule)

Lexical Rule (words)

ngle Rule example taken.

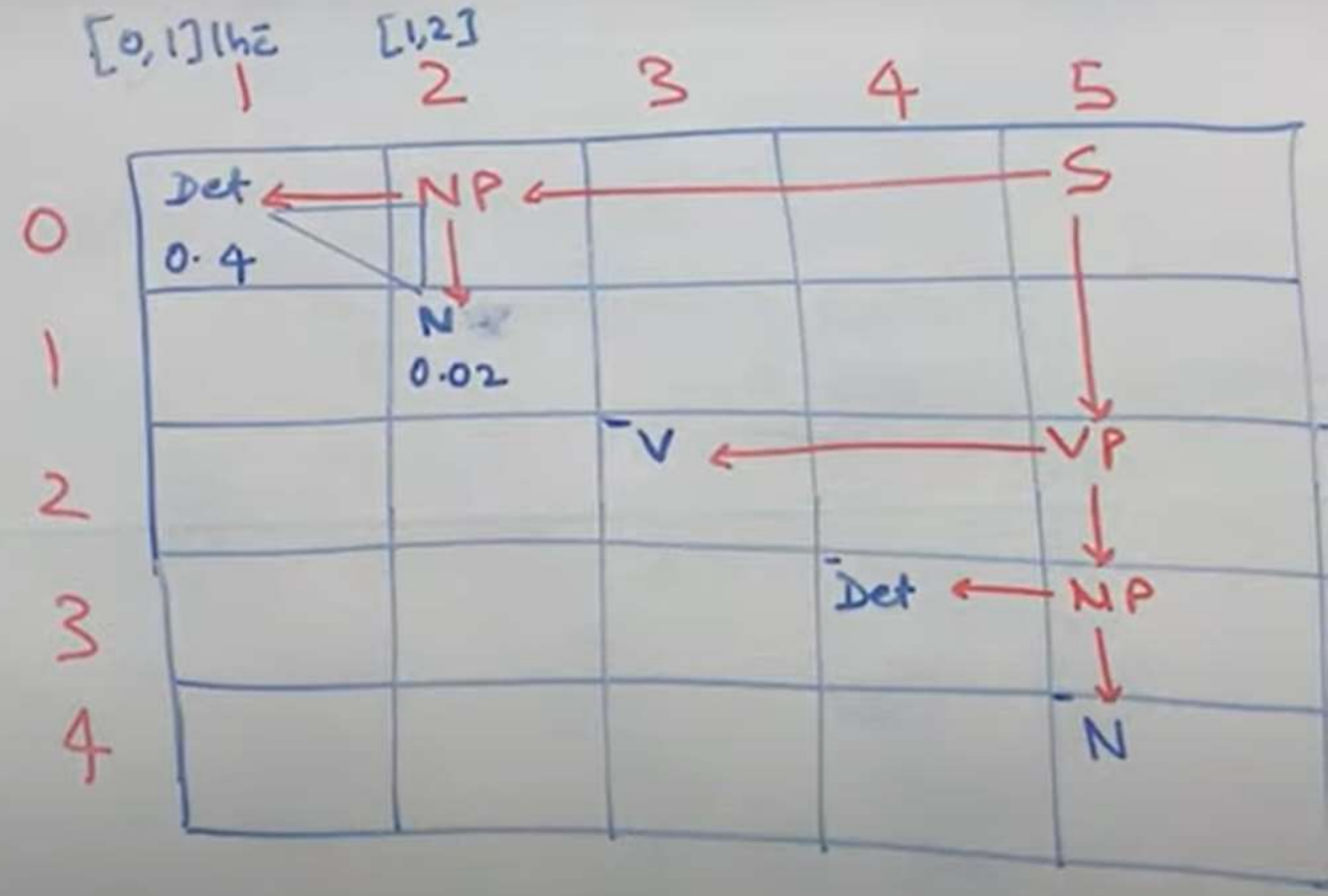


0.3 * 0.4 * 0.02

0.0024

0 lhē 1 flight 2 includes 3 4

5x5 matrix/chart/table



SUBSCRIBE

Models for Ambiguity Resolution in Parsing

- Here we discuss on modelling aspects of parsing: how to design features and ways to resolve ambiguity in parsing.

Probabilistic context-free grammar

- Ex: *John bought a shirt with pockets*

```
(S (NP John)
   (VP (VP (V bought)
           (NP (D a)
               (N shirt)))
       (PP (P with)
           (NP pockets))))
```

```
(S (NP John)
   (VP (V bought)
       (NP (NP (D a)
               (N shirt))
           (PP (P with)
               (NP pockets)))))
```

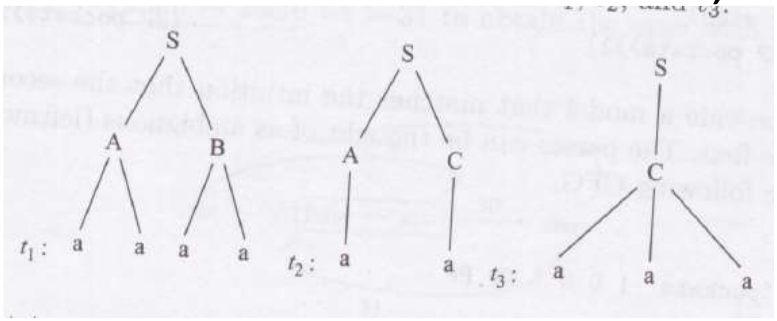
- Here we want to provide a model that matches the intuition that the second tree above is preferred over the first.
- The parses can be thought of as ambiguous (leftmost to rightmost) derivation of the following CFG:

```
S -> NP VP
NP -> 'John' | 'pockets' | D N | NP PP
VP -> V NP | VP PP
V -> 'bought'
D -> 'a'
N -> 'shirt'
PP -> P NP
P -> 'with'
```

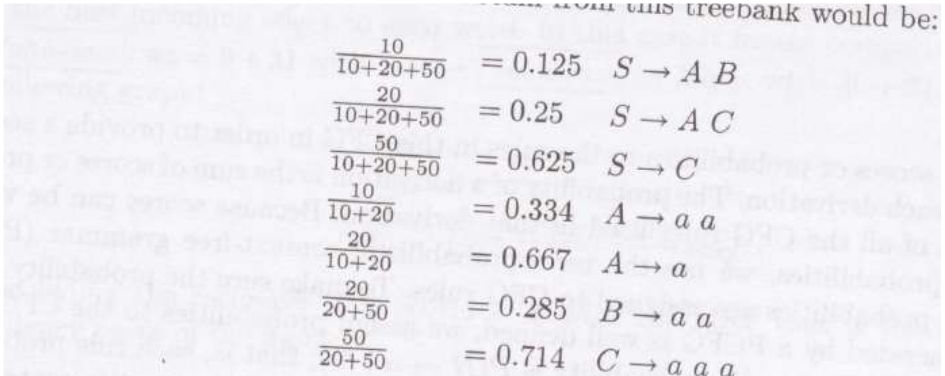
- We assign scores or probabilities to the rules in CGF in order to provide a score or probability for each derivation.

```
S -> NP VP (1.0)
NP -> 'John' (0.1) | 'pockets' (0.1) | D N (0.3) | NP PP (0.5)
VP -> V NP (0.9) | VP PP (0.1)
V -> 'bought' (1.0)
D -> 'a' (1.0)
N -> 'shirt' (1.0)
PP -> P NP (1.0)
P -> 'with' (1.0)
```

- From these rule probabilities, the only deciding factor for choosing between the two parses for John brought a shirt with pockets in the two rules NP->NP PP and VP-> VP PP. The probability for NP -> NP PP is set higher in the preceding PCFG.
- The rule probabilities can be derived from a treebank, consider a treebank with three tress t_1, t_2, t_3



- if we assume that tree t1 occurred 10 times in the treebank, t2 occurred 20 times and t3 occurred 50 times, then the PCFG we obtain from this treebank is:



...from this treebank would be:

$$\frac{10}{10+20+50} = 0.125 \quad S \rightarrow A\ B$$

$$\frac{20}{10+20+50} = 0.25 \quad S \rightarrow A\ C$$

$$\frac{50}{10+20+50} = 0.625 \quad S \rightarrow C$$

$$\frac{10}{10+20} = 0.334 \quad A \rightarrow a\ a$$

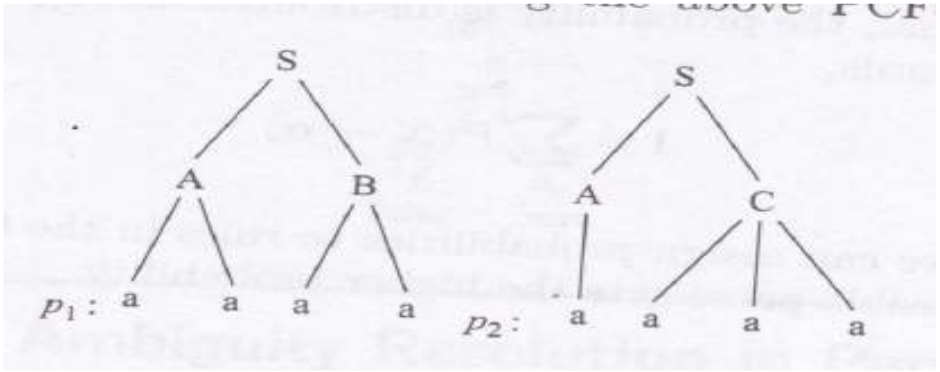
$$\frac{20}{10+20} = 0.667 \quad A \rightarrow a$$

$$\frac{20}{20+50} = 0.285 \quad B \rightarrow a\ a$$

$$\frac{50}{20+50} = 0.714 \quad C \rightarrow a\ a\ a$$

put a a ...

- For input a a a there are two parses using the above PCFG: the probability P1 =0.125 0.334 0.285 = 0.01189 p2=0.25 0.667 0.714 =0.119.
- The parse tree p2 is the most likely tree for that input.



Generative models

- To find the most plausible parse tree, the parser has to choose between the possible derivations each of which can be represented as a sequence of decisions.
- Let each derivation D = $d_1, d_2, \ldots, d_n$, which is the sequence of decisions used to build the parse tree.
- Then for input sentence x, the output parse tree y is defined by the sequence of steps in the derivation.
- The probability for each derivation:

$$P(x, y) = P(d_1, \ldots, d_n) = \prod_{i=1}^{n} P(d_i \mid d_1, \ldots, d_{i-1})$$

- The conditioning context in the probability $P(d_i|d_1,\ldots\ldots,d_{i-1})$ is called the history and corresponds to a partially built parse tree (as defined by the derived sequence).
- We make a simplifying assumption that keeps the conditioning context to a finite set by grouping the histories into equivalence classes using a function

$$P(d_1,\ldots,d_n)=\prod_{i=1}^{n}P(d_i \mid \Phi(d_1,\ldots,d_{i-1}))$$

Using Φ, each history $H_i = d_1,\ldots,d_{i-1}$ for all x, y is mapped to some fixed finite set of feature functions of the history $\phi_1(H_i),\ldots,\phi_k(H_i)$. In terms of these k feature functions:

$$P(d_1,\ldots,d_n)=\prod_{i=1}^{n}P(d_i \mid \phi_1(H_i),\ldots,\phi_k(H_i))$$

Discriminative models for Parsing

- Colins created a simple notation and framework that describes various discriminative approaches to learning for parsing.
- This framework is called global linear model.
- Let x be a set of inputs and y be a set of possible outputs that can be a sequence of POS tags or a parse tree or a dependency analysis.
- Each $x \in x$ and $y \in y$ is mapped to a d-dimensional feature vector ø(x,y), with each dimension being a real number.
- A weight parameter vector $w \in R^d$ assigns a weight to each feature in ø(x,y), representing the importance of that feature.

- The value of ø(x,y).w is the score of (x,y) . The height the score, the more possible it is that y is the output of x.
- The function GEN(x) generates the set of possible outputs y for a given x.
- Having ø(x,y).w and GEN(x) specified, we would like to choose the height scoring candidate y^* from GEN(x) as the most possible output

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}} \; p(y \mid x, \mathbf{w})$$

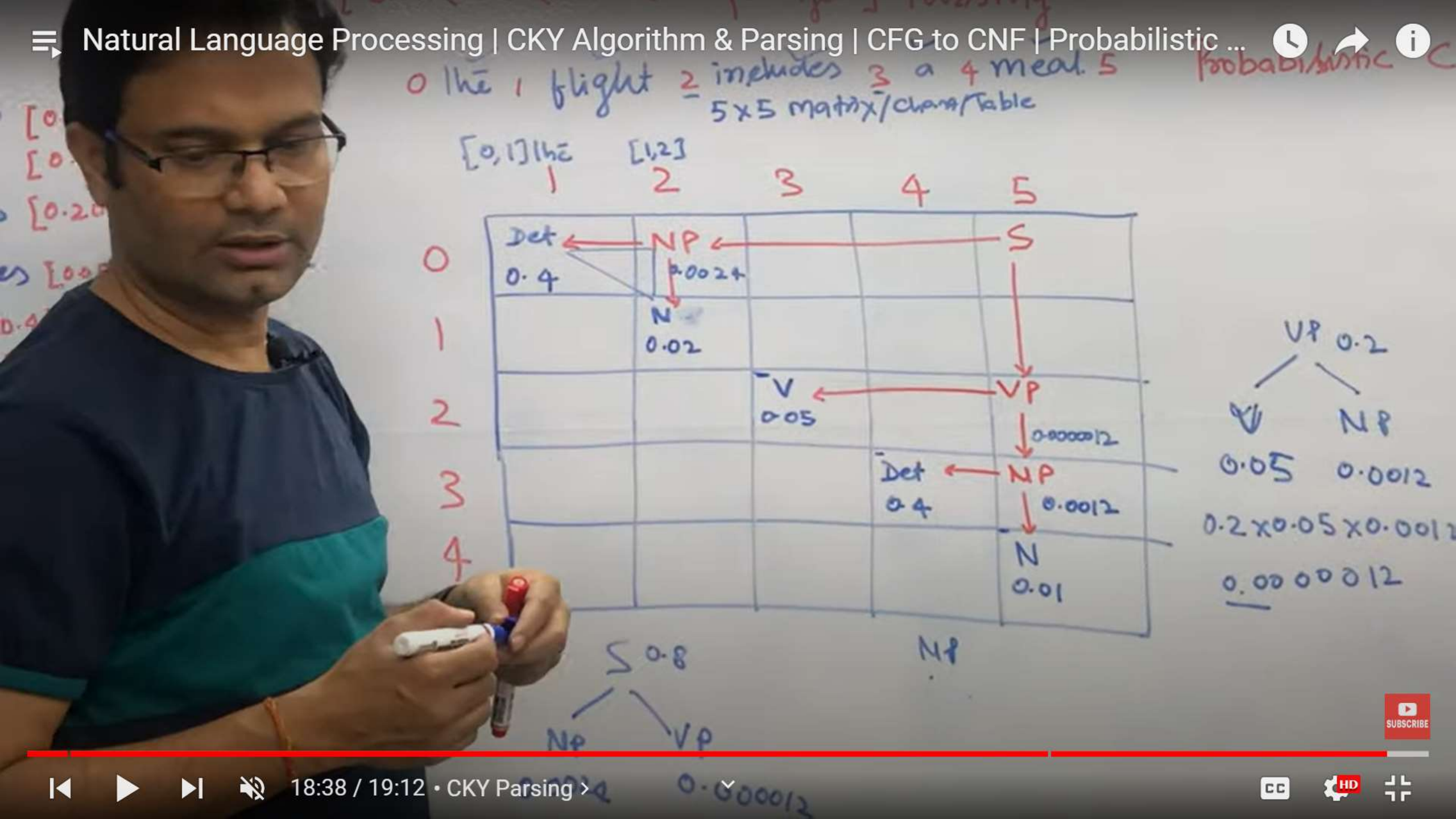
where F(x) returns the highest scoring output y^* from GEN(x)

- A conditional random field (CRF) defines the conditional probability as a linear score for each candidate y and a global normalization term:

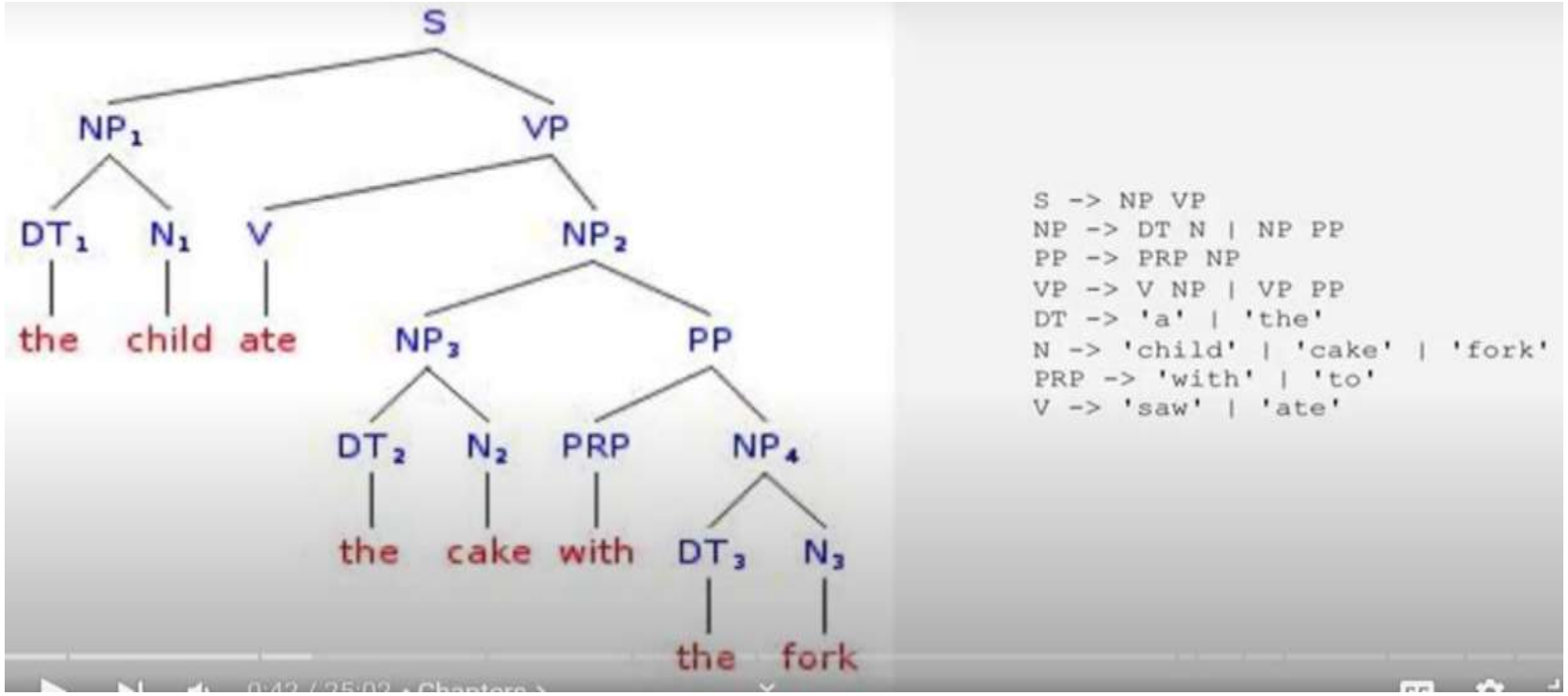
$$\log p(y \mid x, \mathbf{w}) = \Phi(x, y) \cdot \mathbf{w} - \log \sum_{y' \in GEN(x)} \exp(\Phi(x, y') \cdot \mathbf{w})$$

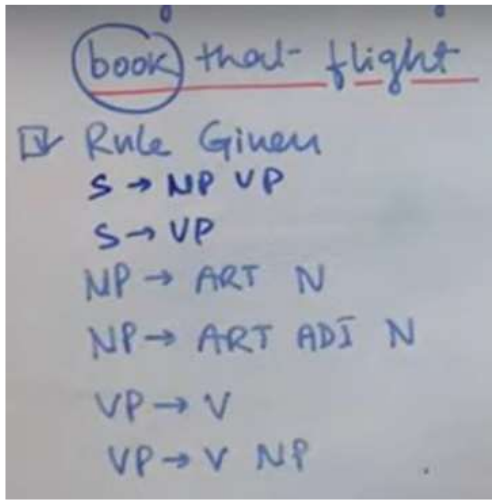
- A simple linear model that ignores the normalization term is:

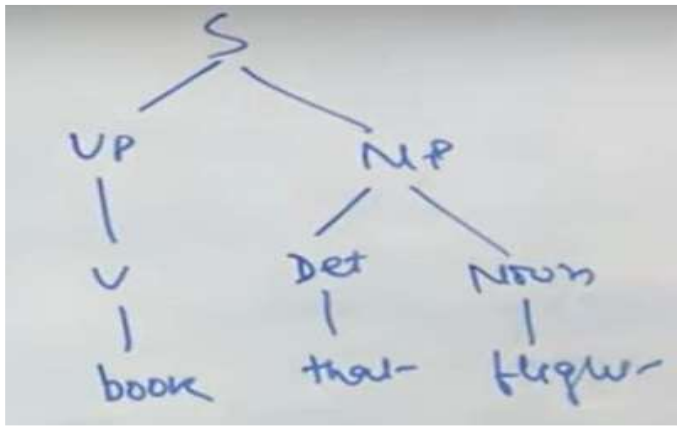
$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}} \; \Phi(x, y) \cdot \mathbf{w}$$

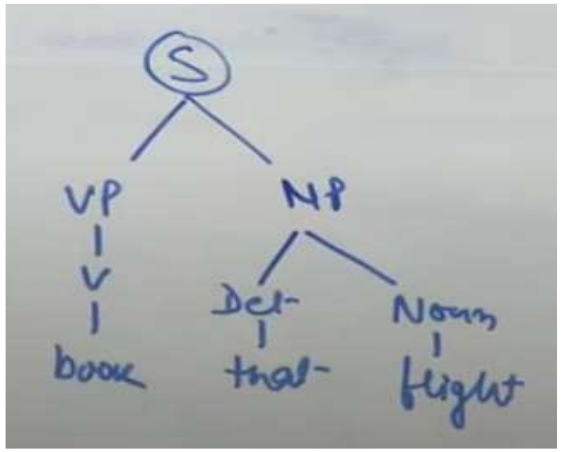

Natural Language Processing | CKY Algorithm & Parsing | CFG to CNF | Probabilistic ...
0 the 1 flight 2 includes 3 a 4 meal 5
5x5 matrix/chart/table
[0,1] the
[1,2]
1
2
3
4
5
0
1
2
3
4
Det
0.4
NP
0.0024
S
N
0.02
V
0.05
VP
0.000012
Det
0.4
NP
0.0012
N
0.01
VP 0.2
V
NP
0.05
0.0012
0.2x0.05x0.001
0.00 00 012
S 0.8
NP
VP
0.00012
18:38 / 19:12 • CKY Parsing

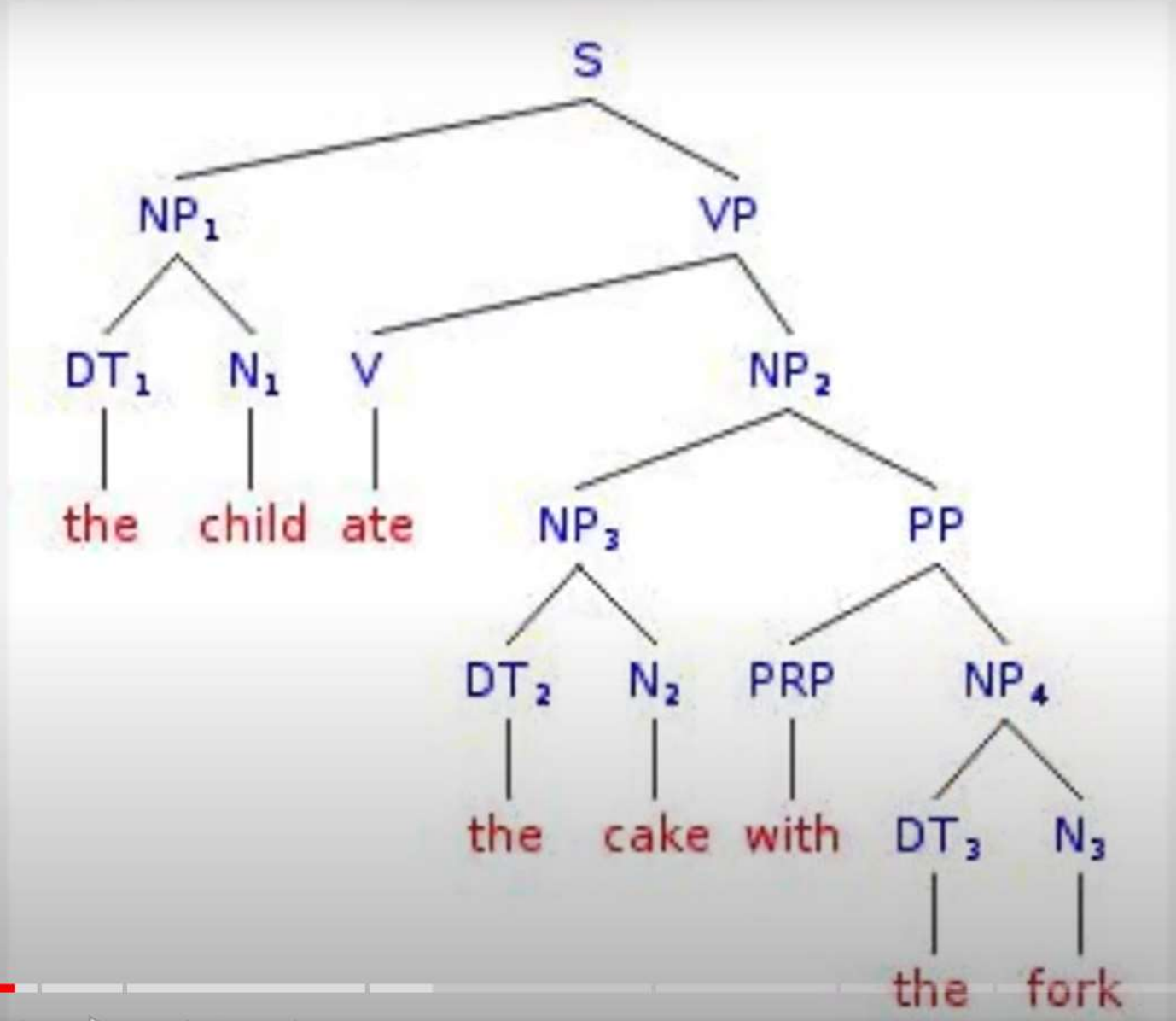
- **There are two general approaches to parsing 1.Top down parsing (start with start symbol) 2.Botttom up parsing (start from terminals)**












```
S -> NP VP
NP -> DT N | NP PP
PP -> PRP NP
VP -> V NP | VP PP
DT -> 'a' | 'the'
N -> 'child' | 'cake' | 'fork'
PRP -> 'with' | 'to'
V -> 'saw' | 'ate'
```

3.a) Explain various types of parsers in NLP.
b) Discuss multilingual issues in detail. [8+7]

4.a) Given Grammar
S→ AB | BB
A→ CC | AB | a
B→ BB | CA | b
C→ BA | AA | b
Word w= aabb. Apply Top Down Parsing test, the word can be generated or not.
b) Explain Tree Banks and its role in parsing. [8+7]

5.a) State the types of references in discourse.

2.a) Elaborate the Models for Ambiguity Resolution in Parsing.
b) With the help of a neat diagram, explain the Representation of Syntactic Structure. [8+7]

R23 SET-2

Semantic Parsing:

Introduction, Semantic Interpretation, System Paradigms, Word Sense Systems, Software.

Semantic Parsing: Introduction

- In language understanding is the identification of a meaning representation that is detailed enough to allow reasoning system to make deduction(the process of reaching a decision or answer by thinking about the known facts).
- But at the same time, is general enough that it can be used across many domains with little to no adaptation(not capable of adjusting to new conditions or situations).
- It is not clear whether a final, low-level, detailed semantics representation covering various applications that use some form of language interface can be achieved or
- An ontology(**a branch of metaphysics concerned with the nature and relations of being**) can be created that can be created that can capture the various granularities and aspects of meaning that are embodied in such that a variety of applications.
- None of these approaches are not created, So two compromise approaches have emerged in the NLP for language understanding.
- In the first approach, a specific, rich meaning representation is created for a limited domain for use by application that are restricted to that domain, such as travel reservations, football game simulations, or querying a geographic database.

- In the second approach, a related set of intermediate-specific meaning representation is created, going from low-level analysis to a middle analysis, and the bigger understanding task is divided into multiple, smaller pieces that are more manageable, such as word sense disambiguation followed by predicate-argument structure recognition.
- Here two types of meaning representations: a domain-dependent, deeper representation and a set of relatively shallow but general-purpose, low-level, and intermediate representation.
- The task of producing the output of the first type is often called deep semantic parsing, and the task of producing the output of the second type is often called shallow semantic parsing.
- The first approach is so specific that porting to every new domain can require anywhere from a few modifications to almost reworking the solution from scratch.
- In other words, the reusability of the representation across domains is very limited.
- The problem with second approach is that it is extremely difficult to construct a general purpose ontology and create symbols that are shallow enough to be learnable but detailed enough to be useful for all possible applications.
- Therefore, an application specific translation layer between the more general representation and the more specific representation becomes necessary.

2.Semantic Interpretation

- Semantic parsing can be considered as part of Semantic interpretation, which involves various components that together define a representation of text that can be fed into a computer to allow further computations manipulations and search, which are prerequisite for any language understanding system or application. Here we start with discus with structure of semantic theory.
- A Semantic theory should be able to:
  1. Explain sentence having ambiguous meaning: The bill is large is ambiguous in the sense that is could represent money or the beak of a bird.
  2. Resolve the ambiguities of words in context. The bill is large but need not be paid, the theory should be able to disambiguate the monetary meaning of bill.
  3. Identify meaningless but syntactically well-formed sentence: Colorless green ideas sleep furiously.
  4. Identify syntactically or transformationally unrelated paraphrasers of concept having the same semantic content.

- Here we look at some requirements for achieving a semantic representation

2.1.Structural ambiguity

- Structure means syntactic structure of sentences.
- The syntactic structure means transforming the a sentence into its underlying syntactic representation and in theory of semantic interpretation refer to underlying syntactic representation.

2.2.Word Sense

- In any given language, the same word type is used in different contexts and with different morphological variants to represent different entities or concepts in the world.
- For example, we use the word **nail** to represent a part of the human anatomy and also to represent the generally metallic object used to secure other objects.

intended by the author or speaker. Let's take the following four examples. The presence of words such as *hammer* and *hardware store* in sentences 1 and 2, and of *clipped* and *manicure* in sentences 3 and 4, enable humans to easily disambiguate the sense in which *nail* is used:

1. He *nailed* the loose arm of the chair with a hammer.
2. He bought a box of *nails* from the hardware store.
3. He went to the beauty salon to get his *nails* clipped.
4. He went to get a manicure. His *nails* had grown very long.

Resolving the sense of words in a discourse, therefore, constitutes one of the steps in the process of semantic interpretation. We discuss it in greater depth in Section 4.4.

2.3.Entity and Event Relation

- The next important component of semantic interpretation is the identification of various entities that are sparkled across the discourse using the same or different phrases.
- The predominant tasks have become popular over the years: **named entity recognition** and **coreference resolution**.

2.4.Predicate-Argument Structure

- Once we have the word-sense, entities and events identified, another level of semantics structure comes into play: identifying the participants of the entities in these events.
- Resolving the argument structure of predicate in the sentence is where we identify which entities play what part in which event.
- A word which functions as the verb does here, we call a **predicate** and words which function as the nouns do are called **arguments**. Here are some other predicates and arguments:
- Selena slept
  argument predicate
  Tom is tall
  argument predicate
  Percy placed the penguin on the podium
  argument predicate argument argument
- "Sanchon serves vegetarian food" can be described in FOPC as: Server (Sanchon, VegetarianFood)
- Generally, this process can be defined as the identification of who did what to whom, where, why and how.

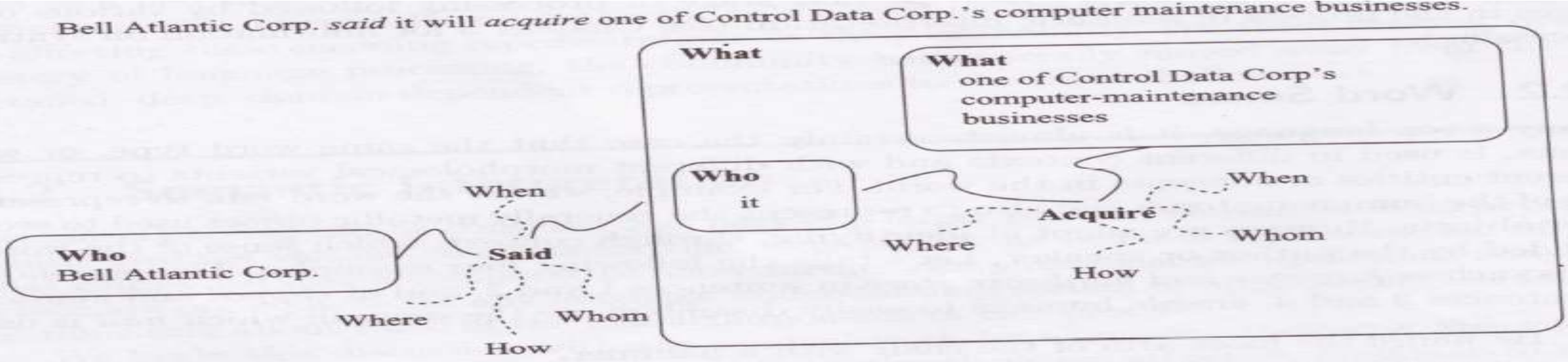


Figure 4–1: A representation of *who* did *what* to *whom*, *when*, *where*, *why*, and *how*

2.5.Meaning Representation

- The final process of the semantic interpretation is to build a semantic representation or meaning representation that can then be manipulated by algorithms to various application ends.
- This process is sometimes called the deep representation.
- The following two examples

(1) If our player 2 has the ball, then position our player 5 in the midfield.

```
((bowner (player our 2)) (do (player our 5) (pos (midfield)))
```

(2) Which river is the longest?

```
answer(x1, longest(x1 river(x1)))
```

3.System Paradigms

- It is important to get a perspective on the various primary dimensions on which the problem of semantic interpretation has been tackled.
- The approaches generally fall into the following three categories: 1.System architecture 2.Scope 3. Coverage.
- **System Architectures**

a.Knowledge based: These systems use a predefined set of rules or a knowledge base to obtain a solution to a new problem.

b.**Unsupervised**: These systems tend to require minimal human intervention to be functional by using existing resources that can be bootstrapped for a particular application or problem domain.

c.Supervised: these systems involve the manual annotation of some phenomena that appear in a sufficient quantity of data so that machine learning algorithms can be applied.

d.Semi-Supervised: manual annotation is usually very expensive and does not yield enough data to completely capture a phenomenon. In such instances, researches can automatically expand the data set on which their models are trained either by employing machine-generated output directly or by bootsrapping off an existing model by having humans correct its output.

2.**Scope**:

a.**Domain Dependent**: These systems are specific to certain domains, such as air travel reservations or simulated football coaching.

b.**Domain Independent**: These systems are general enough that the techniques can be applicable to multiple domains without little or no change.

3.**Coverage**:

a.**Shallow**: These systems tend to produce an intermediate representation that can then be converted to one that a machine can base its action on.

b. **Deep**: These systems usually create a terminal representation that is directly consumed by a machine or application.

4.Word Sense

- **Word Sense** Disambiguation is an important method of **NLP** by which the meaning of a word is determined, which is used in a particular context.
- In a compositional approach to semantics, where the meaning of the whole is composed on the meaning of parts, the smallest parts under consideration in textual discourse are typically the words themselves: either tokens as they appear in the text or their lemmatized forms.
- Words sense has been examined and studied for a very long time.
- Attempts to solve this problem range from rule based and knowledge based to completely unsupervised, supervised, and semi-supervised learning methods.
- Very early systems were predominantly **rule based or knowledge based** and used dictionary definitions of senses of words.
- **Unsupervised** word sense induction or disambiguation techniques try to induce the senses or word as it appears in various corpora.
- These systems perform either a hard or soft clustering of words and tend to allow the tuning of these clusters to suit a particular application.
- Most recent **supervised** approaches to word sense disambiguation, usually application-independent-level of granularity (including small details). Although the output of supervised approaches can still be amendable to generating a ranking, or distribution, of membership sense

- Word sense ambiguities can be of three principal types: i.**homonymy** ii.**polysemy** iii.**categorial ambiguity**.
- **Homonymy** defined as the words having same spelling or same form but having different and unrelated meaning. For example, the word “Bat” is a homonymy word because bat can be an implement to hit a ball or bat is a nocturnal flying mammal also
- **Polysemy** is a Greek word, which means “many signs”. polysemy has the same spelling but different and related meaning.
- Both polysemy and homonymy words have the same syntax or spelling. The main difference between them is that in polysemy, the meanings of the words are related but in homonymy, the meanings of the words are not related.
- For example: Bank Homonymy: financial bank and river bank
  Polysemy: financial bank, bank of clouds and book bank: indicate collection of things.
- **Categorial ambiguity**: the word book can mean a book which contain the chapters or police register which is used to enter the charges against some one.
  text book and note book
- In the above note book belongs to the grammatical category of noun, and text book is verb.

- Distinguishing between these two categories effectively helps disambiguate these two senses.
- Therefore categorical ambiguity can be resolved with syntactic information (part of speech) alone, but polyseme and homonymy need more than syntax.
- Traditionally, in English, word senses have been annotated for each part of speech separately, whereas in Chinese, the sense annotation has been done per lemma.

Resources

- As with any language understanding task, the availability of resources is key factor in the disambiguation of the wors senses in corpora.
- Early work on wors sense disambiguation used machine readable dictionaries or thesaususes as knowledge sources.
- Two prominent sources were the Longman dictionary of contemporary English (LDOCE) and Roget's Thesaurus.
- The biggest sense annotation corpus OntoNotes released through Lissuistic Data Consortium (LDC).
- The Chinese annotation corpus is HowNet.

Systems

- Researchers have explored various system architectures to address the sense disambiguation problem.
- We can classify these systems into four main categories: i. rules based or knowledge ii. Supervised iii.unsupervised iv. Semisupervised

Rule Based:

- The first generation pf word sense disambiguation systems was primarily based on dictionary sense definitions.
- Much of this information is historical and cannot readily be translated and made available for building systems today. But some of techniques and algorithms are still available.
- The simplest and oldest dictionary based sense disambiguation algorithm was introduced by leak.
- The core of the algorithm is that the dictionary sense whose terms most closely overlap with the terms in the context.

Algorithm 4–1 Pseudocode of the simplified Lesk algorithm
The function COMPUTEOVERLAP returns the number of words common to the two sets

Procedure: SIMPLIFIED_LESK(word, sentence) **returns** best sense of *word*

```
1:  best-sense ← most frequent sense of word
2:  max-overlap ← 0
3:  context ← set of words in sentence
4:  for all sense ∈ senses of word do
5:      signature ← set of words in gloss and examples of sense
6:      overlap ← COMPUTEOVERLAP(signature, context)
7:      if overlap gt max-overlap then
8:          max-overlap ← overlap
9:          best-sense ← sense
10:     end if
11: end for
12: return best-sense
```

The Simplified Lesk algorithm

- Let's disambiguate "**bank**" in this sentence:

  The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

- given the following two WordNet senses:

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

Choose sense with most word overlap between gloss and context
(not counting function words)

The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

- Another dictionary based algorithm was suggested Yarowsky.
- This study used Roget's Thesaurus categories and classified unseen words into one of these 1042 categories based on a statistical analysis of 100 word concordances for each member of each category.
- The method consists of three steps, as shown in Fig.
- The first step is a collection of contexts.
- The second step computes weights for each of the salient words.
- P(w|Rcat) is the probability of a word w occurring in the context of a Roget's Thesaurus category Rcat.
- P(w|Rcat) |Pr(w) , the probability of a word (w) appearing in the context of a Roget category divided by its overall probability in the corpus.
- Finally, in third step, the unseen words in the test set are classified into the classified into the category that has the maximum weight.

1. Collect contexts for each of the *Roget's Thesaurus* categories.
2. Determine weights for each of the salient words in the context.

$$\frac{P(w_i|RCat)}{P(w_i)}$$

3. Use the weights for predicting the appropriate category of the word in the test corpus.

$$\arg\max_{RCat} \sum_{W} \log \frac{P(w_i|RCat)P(RCat)}{P(w_i)}$$

Figure 4–2: Algorithm for disambiguating words into *Roget's Thesaurus* categories

Supervised

- The simpler form of word sense disambiguating systems the supervised approach, which tends to transfer all the complexity to the machine learning machinery while still requiring hand annotation tends to be superior to unsupervised and performs best when tested on annotated data.
- These systems typically consist of a machine learning classifier trained on various features extracted for words that have been manually disambiguated in a given corpus and the application of the resulting models to disambiguating words in the unseen test sets.
- A good feature of these systems is that the user can incorporate rules and knowledge in the form of features.

Classifier:

- Probably the most common and high performing classifiers are support vector machine (SVMs) and maximum entropy classifiers.

Features: Here we discuss a more commonly found subset of features that have been useful in supervised learning of word sense.

- **Lexical context**: The feature comprises the words and lemma of words occurring in the entire paragraph or a smaller window of usually five wors.
- **Parts of speech** : the feature comprises the POS information for words in the window surrounding the word that is being sense tagged.

- **Bag of words context**: this feature comprises using an unordered set of words in the context window.
- **Local Collections** : Local collections are an ordered sequence of phrases near the target word that provide semantic context for disambiguation. Usually, a very small window of about three tokens on each side of the target word, most often in contiguous pairs or triplets, are added as a list of features.
- **Syntactic relations**: if the parse of the sentence containing the target word is available, then we can use syntactic features.
- **Topic features**: The board topic, or domain, of the article that word belongs to is also a good indicator of what sense of the word might be most frequent.

is also a good indicator of what sense of the word might be most frequent.

Chen and Palmer [51] recently proposed some additional rich features for disambiguation:

Voice of the sentence—This ternary feature indicates whether the sentence in which the word occurs is a passive, semipassive,[3] or active sentence.

Presence of subject/object—This binary feature indicates whether the target word has a subject or object. Given a large amount of training data, we could also use the actual lexeme and possibly the semantic roles rather than the syntactic subject/objects.

Sentential complement—This binary feature indicates whether the word has a sentential complement.

Prepositional phrase adjunct—This feature indicates whether the target word has a prepositional phrase, and if so, selects the head of the noun phrase inside the prepositional phrase.

Named entity—This feature is the named entity of the proper nouns and certain types of common nouns.

WordNet—WordNet synsets of the hypernyms of head nouns of the noun phrase arguments of verbs and prepositions.

[illegible] following research in semantic role labeling. Dligach and Palmer [52]

Unsupervised

Progress in word sense disambiguation is stymied by the dearth of labeled training data to train a classifier for every sense of each word in a given language. There are a few solutions to this problem:

1. Devise a way to cluster instances of a word so that each cluster effectively constrains the examples of the word to a certain sense. This could be considered **sense induction** through clustering.
2. Use some metrics to identify the proximity of a given instance with some sets of known senses of a word and select the closest to be the sense of that instance.
3. Start with **seeds** of examples of certain senses, then iteratively grow them to form clusters.

We first look at the category of algorithms that use some form of distance measure to identify senses. Rada et al. [53] introduced a metric for computing the shortest distance between the two pairs of senses in WordNet. This metric assumes that multiple co-occurring words are likely to exhibit senses that would minimize the distance in a semantic network of hierarchical relations, for example, IS-A, from WordNet. Resnik [54] proposed a new measure of semantic similarity: **information content** in an IS-A taxonomy which produces much better results than the edge-counting measure. Agirre and Rigau [55] further refined this measure, calling it **conceptual density**, which not only depends on the number of separating edges but is also sensitive to the depth of the hierarchy and the density of its concepts and is independent of the number of concepts being measured. Conceptual density is defined for each of the subhierarchies in Figure 4–4. The sense that

4.4.3 Software

Several software programs are made available by the research community for word sense disambiguation, ranging from similarity measure modules to full disambiguation systems. It is not possible to list all of them here, so we list a selected few.

- **IMS** (It Makes Sense) http://nlp.comp.nus.edu.sg/software
  This is a complete word sense disambiguation system.
- **WordNet-Similarity-2.05** http://search.cpan.org/dist/WordNet-Similarity
  These WordNet Similarity modules for Perl provide a quick way of computing various word similarity measures.
- **WikiRelate!**
  http://www.h-its.org/english/research/nlp/download/wiki pediasimilarity.php
  This is a word similarity measure based on the categories in Wikipedia.

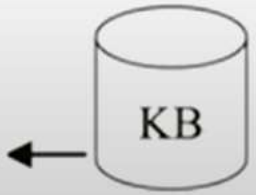
Meaning Representation for Natural Language Understanding - Mariana Romanyshyn | - ODSC ...

Lambda calculus

Sentence: *What is the capital of Germany?*
Which city is the capital of Germany?
Which city in Germany is the capital?

Logical form: λx.capital(x, Germany)

Result:

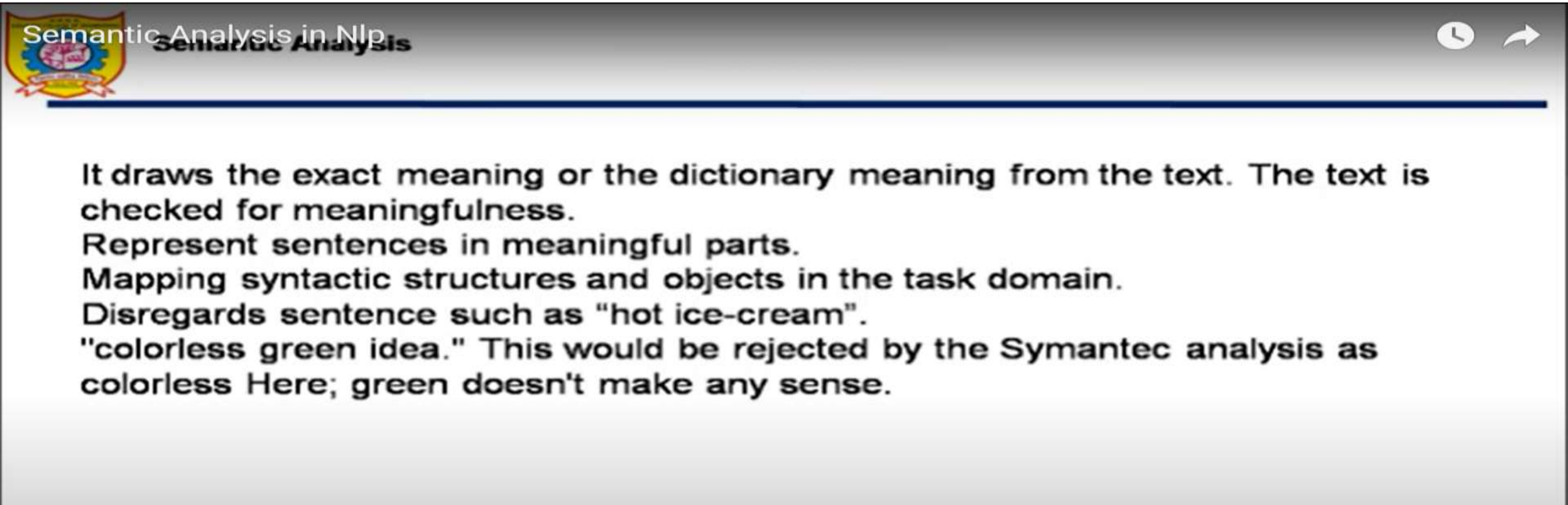


19

ODSC

9:33 / 44:19 • Lambda calculus ›

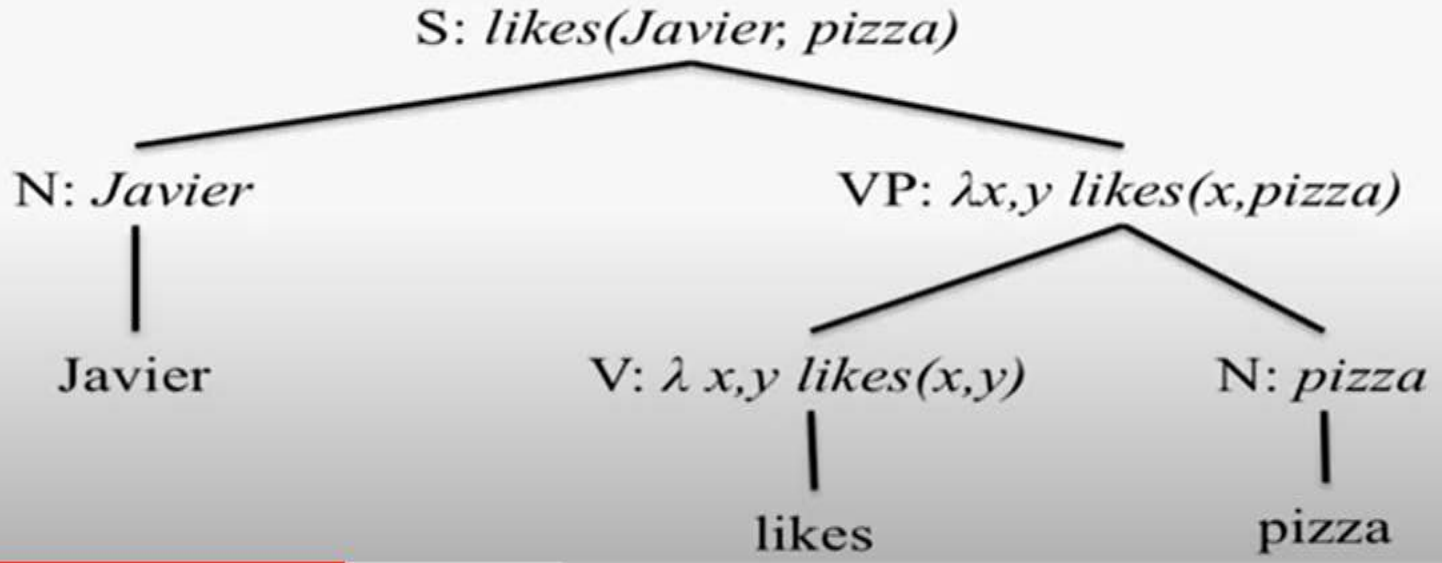
- Semantic analysis starts with lexical semantics, which studies individual words' meanings (i.e., dictionary definitions).
- Semantic analysis then examines relationships between individual words and analyzes the meaning of words that come together to form a sentence.
- This analysis provides a clear understanding of words in context. For Example:
  "The boy ate the apple" defines an apple as a fruit.
  "The boy went to Apple" defines Apple as a brand or store.



Press Esc to exit full screen

Semantic Parsing

- Associate a semantic expression with each node

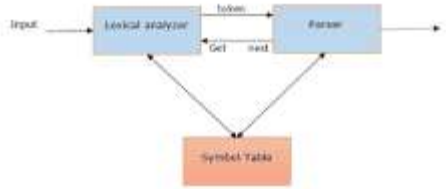


2:40 / 9:17

https://www.youtube.com/watch?v=eEjU8oY_7DE
https://www.youtube.com/watch?v=W7QdqCrX_mY
https://www.youtube.com/watch?v=XLvv_5meRNM
https://www.geeksforgeeks.org/understanding-semantic-analysis-nlp/#:~:text=Semantic%20Analysis%20is%20a%20subfield,process%20to%20us%20as%20humans.

- What is deep parsing in NLP?

In deep parsing, **the search strategy will give a complete syntactic structure to a sentence**. It is the task of parsing a limited part of the syntactic information from the given task. It is suitable for complex NLP applications. It can be used for less complex NLP applications.



- How can semantics be represented in natural language processing systems?

The semantics, or meaning, of an expression in natural language can be **abstractly represented as a logical form**. Once an expression has been fully parsed and its syntactic ambiguities resolved, its meaning should be uniquely represented in logical form.

- What is meant by semantic translation?

Semantic translation is **the process of using semantic information to aid in the translation of data in one representation or data model to another representation or data model**.

- Semantic meaning can be studied at several different levels within linguistics. The three major types of semantics are **formal, lexical, and conceptual semantics**

Categories of Semantics

Nick Rimer, author of *Introducing Semantics*, goes into detail about the two categories of semantics. "Based on the distinction between the meanings of words and the meanings of sentences, we can recognize two main divisions in the study of semantics: *lexical semantics* and *phrasal semantics*. Lexical semantics is the study of word meaning, whereas phrasal semantics is the study of the principles which govern the construction of the meaning of phrases and of sentence meaning out of compositional combinations of individual lexemes.

A bird's bill, also called **a beak**, **A bird's horny projecting jaws; a bill**

Unit-IV

Predicate Argument Structure:
Predicate Argument Structure
Meaning Representation Systems
Software

Predicate Argument Structure

- Shallow semantics parsing or semantic role labelling, is the process of identifying the various arguments of predicates in a sentence.
- In linguistics, *predicate* refers to the main verb in the sentence. Predicate takes arguments.
- The role of **Semantic Role Labelling (SRL)** is to determine how these arguments are semantically related to the predicate.
- Consider the sentence "Mary loaded the truck with hay at the depot on Friday".
- Loaded' is the predicate. Mary, truck and hay have respective semantic roles of loader, bearer and cargo. Mary->loader, truck->bearer hay->cargo
- We can identify additional roles of location (depot) and time (Friday).
- The job of SRL is to identify these roles so that NLP tasks can "understand" the sentence.
- Often an idea can be expressed in multiple ways. Consider these sentences that all mean the same thing: "Yesterday, Kristina hit Scott with a baseball"; "Scott was hit by Kristina yesterday with a baseball"; "With a baseball, Kristina hit Scott yesterday"; "Kristina hit Scott with a baseball yesterday".
- Either constituent or dependency parsing will analyze these sentence syntactically. But syntactic relations don't necessarily help in determining semantic roles.

- Who hit Scott with a baseball?
- Whom did Kristina hit with a baseball?
- What did Kristina hit Scott with?
- When did Kristina hit Scott with a baseball?

- SRL is useful in any NLP application that requires semantic understanding: machine translation, information extraction, text summarization, question answering, and more.

Resources:

- The late 1990s saw the emergence of two important corpora that are semantically tagged, one is FrameNet and the other is PropBank.
- These resources have begun a transition from a long tradition of predominantly rule-based approaches toward more data-oriented approaches.
- These approaches focus on transforming linguistic insights into features, rather than into rules and letting a machine learning framework use those features to learn a model that helps automatically tag the semantic information encoded in such resources.
- FrameNet is based on the theory of frame semantics, where a given predicate invokes a semantic frame, this instantiating some or all of the possible semantic roles belonging to that frame.

- PropBank on the, on the other hand, is based on Dowty's prototype theory and uses a more linguistically neutral view in which each predicate has a set of core arguments that are predicate dependent and all predicates share a set of non-core or adjunctive, arguments.

FrameNet

- FrameNet contains frame-specific semantic annotation of a number of predicates in English.
- It contains tagged sentences extracted from British National Corpus (BNC).
- The process of FrameNet annotation consists of identifying specific semantic frames and creating a set of frame-specific roles called frame elements.
- Then, a set of predicates that instantiate the semantic frame, irrespective of their grammatical category, are identified, and a variety of sentences are labelled for those predicates.
- The labelling process entails identifying the frame that an instance of the predicate lemma invokes, then identifying semantic arguments for that instance, and tagging them with one of the predetermined ser of frame elements for that frame.
- The combination of the predicate lemma and the frame that its instance invokes is called a lexical unit (LU).
- This is therefore the pairing of a word with its meaning.
- Each sense of a polysemous word tends to be associated with a unique frame.

sense of a polysemous word tends to be associated with a unique frame. For example, the verb *break* can mean *fail to observe (a law, regulation, or agreement)* and can belong to a COMPLIANCE frame along with other word meanings such as *violation, obey, flout*; or it can mean *cause to suddenly separate into pieces in a destructive manner* and can belong to a CAUSE_TO_FRAGMENT frame along with other meanings such as *fracture, fragment, smash.*

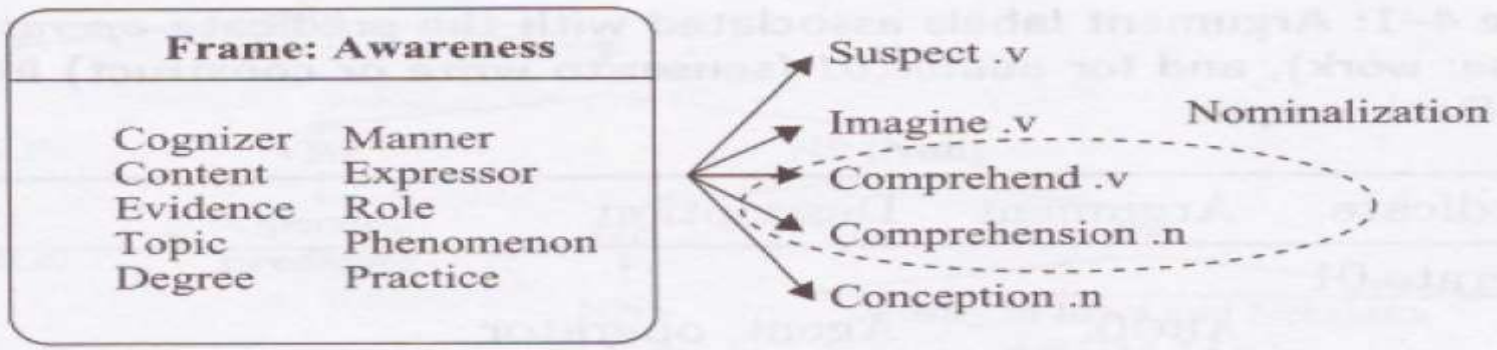


Figure 4–9: FrameNet example

The following example illustrates the general idea. Here, the frame AWARENESS is instantiated by the verb predicate *believe* and the noun predicate *comprehension*. Figure 4–9 shows the AWARENESS frame along with the frame-elements and a sample set of predicates that involve it—including verbs and nominalizations.

1. [*Cognizer* We] [*Predicate:verb believe*] [*Content* it is a fair and generous price]
2. No doubts existed as to [*Cognizer* our] [*Predicate:noun comprehension*] [*Content* of it]

PropBank (propositional Bank)

- PropBank only includes annotation of arguments of verb predicates.
  - Due to the difficulty of defining a universal set of thematic roles, the roles in PropBank are defined w.r.t. each verb sense.
- The arguments are tagged as either core arguments, with labels of the type ARGN, where N takes values from 0 to 5, or adjunctive arguments(listed in table) with labels of the type ARGM-X, where X can take values such as TMP for temporal, LOC for locative and so on.
- Adjunctive arguments share the same meaning across all predicates, where as the meaning of core arguments has to be interpreted in connection with a predicate.

Numbered roles, rather than named roles

- e.g. **Arg0, Arg1, Arg2, Arg3**, and so on

- ARG0 in the PROTO-AGENT (usually the subject of the a transitive verb, ARG1 is the PROTO-PATIENT (usually its direct object of the transitive verb).
- Table 4-1 shows a list of core arguments for the predicates operate and author.
- Note that some core arguments, such as ARG2 and ARG3, do not occur with author.
- This is explained by the fact that not all core arguments can be instantiated by all senses of all predicates.
- A list of core arguments that can occur with a particular sense of the predicate, along with their real-world meaning, is present in a file called the frames file. One frames is associated with each predicate.

Table 4–1: Argument labels associated with the predicate *operate.01* (sense: work), and for *author.01* (sense: to write or construct) in the PropBank corpus

Predicate	Argument	Description
operate.01		
	ARG0	Agent, operator
	ARG1	Thing operated
	ARG2	Explicit patient (thing operated on)
	ARG3	Explicit argument
	ARG4	Explicit instrument
author.01		
	ARG0	Author, agent
	ARG1	Text authored

Table 4–2: List of adjunctive arguments in PropBank—ARGMs

Tag	Description	Examples
ARGM-LOC	Locative	*the museum, in Westborough, Mass.*
ARGM-TMP	Temporal	*now, by next summer*
ARGM-MNR	Manner	*heavily, clearly, at a rapid rate*
ARGM-DIR	Direction	*to market, to Bangkok*
ARGM-CAU	Cause	*In response to the ruling*
ARGM-DIS	Discourse	*for example, in part, Similarly*
ARGM-EXT	Extent	*at $38.375, 50 points*
ARGM-PRP	Purpose	*to pay for the plant*
ARGM-NEG	Negation	*not, n't*
ARGM-MOD	Modal	*can, might, should, will*
ARGM-REC	Reciprocals	*each other*
ARGM-PRD	Secondary Predication	*to become a teacher*
ARGM	Bare ARGM	*with a police escort*
ARGM-ADV	Adverbials	*(none of the above)*

An example extracted from the PropBank corpus along with its syntax tree representation and argument labels is shown in Figure 4–10

PropBank argument numbering

Although numbering differs per **verb sense**, the general pattern of numbering is as follows:

- Arg0 = "Proto-Agent" (agent)
- Arg1 = "Proto-Patient" (direct object / theme / patient)
- Arg2 = indirect object (benefactive / instrument / attribute / end state)
- Arg3 = start point (benefactive / instrument / attribute)
- Arg4 = end point

Different "frameset" for each verb sense

- *Mary left the room*
- *Mary left her daughter-in-law her pearls in her will*

Frameset **leave.01** "move away from":
Arg0: entity leaving
Arg1: place left

Frameset **leave.02** "give":
Arg0: giver
Arg1: thing given
Arg2: beneficiary

Ergative/Unaccusative Verbs

Roles (no ARG0 for unaccusative verbs)

- Arg1 = Logical subject, patient, thing rising
- Arg2 = EXT, amount risen
- Arg3* = start point
- Arg4 = end point

Sales rose 4% to $3.28 billion from $3.16 billion.

The Nasdaq composite index added 1.01 to 456.6 on paltry volume.

Modifiers or adjuncts of the predicate: Arg-M-...

TMP	when?	yesterday evening, now
LOC	where?	at the museum, in San Francisco
DIR	where to/from?	down, to Bangkok
MNR	how?	clearly, with much enthusiasm
PRP/CAU	why?	because ... , in response to the ruling
REC		themselves, each other
ADV	miscellaneous	
PRD	secondary predication	...ate the meat raw

Other Resources

Other resources have been developed to aid further research in predicate-argument recognition. NomBank [78] was inspired by PropBank. In the process of identifying and tagging the arguments of nouns, the NOMLEX (NOMinalization LEXicon) [79] dictionary was expanded to cover about 6,000 entries. Along with this, the frames from PropBank were used to generate the frame files for NomBank. Another resource that ties PropBank frames with more predicate-independent thematic roles and also provides a richer representation associating the framesets with Levin classes [80] is VerbNet [81]. In fact the PropBank frames have a strong connection with the Levin verb classes, specifically the intersective Levin classes [82]. FrameNet frames are also somewhat related in the sense that FrameNet's generation of verb classes is more data driven than theoretical. Baker and Ruppenhofer [83] present an interesting discussion on how the FrameNet frames relate to Levin classes.

Although FrameNet and PropBank started with annotating predicate-argument structures in English, it was not long before their philosophy propagated to other languages. Because FrameNet was based on frame semantics with coarse-grained semantic frames, and the nature of semantics is lingua independent, it was apparent that these frames could be reused to annotate data in other languages. The SALSA project [84, 85] was the first to put this into practice. FrameNet tags both literal and metaphorical interpretations of text, but that can lead to ambiguity and lower consistency, so the SALSA project remained close to the literal meaning. It reused all possible, preexisting FrameNet frames, and when linguistic semantic parallelism did not propagate across languages, they created more frames. Subsequently, there exist FrameNets in Japanese [86, 87], Spanish [88], and Swedish [89]. As of this writing, there are FrameNet projects underway in more than 10 languages [90].

https://towardsdatascience.com/understanding-frame-semantic-parsing-in-nlp-bec84c885061

FrameNet

FrameNet contains frame-specific semantic annotation of a number of predicates in English. It contains tagged sentences extracted from the British National Corpus (BNC). The process of FrameNet annotation consists of identifying specific semantic frames and creating a set of frame-specific roles called **frame elements**. Then, a set of predicates that instantiate the semantic frame, irrespective of their grammatical category, are identified, and a variety of sentences are labeled for those predicates. The labeling process entails identifying the frame that an instance of the predicate lemma invokes, then identifying semantic arguments for that instance, and tagging them with one of the predetermined set of frame elements for that frame. The combination of the predicate lemma and the frame that its instance invokes is called a lexical unit (LU). This is therefore the pairing of a word with its meaning. Each sense of a polysemous word tends to be associated with a unique frame. For example, the verb *break* can mean *fail to observe (a law, regulation, or agreement)* and can belong to a COMPLIANCE frame along with other word meanings such as *violation, obey, flout*; or it can mean *cause to suddenly separate into pieces in a destructive manner* and can belong to a CAUSE_TO_FRAGMENT frame along with other meanings such as *fracture, fragment, smash.*