SITA3012 Natural Language Processing

Unit 1

Introduction

Introduction and challenges of natural language processing, Phases in natural language processing, An outline of English syntax - Grammars and parsing - Features and Augmented Grammar.

1.1 Introduction to NLP:

- Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interaction between computers and humans in natural language.
- Natural language processing studies interactions between humans and computers to find ways for computers to process written and spoken words similar to how humans do.
- Natural language processing (NLP) is the intersection of computer science, linguistics and machine learning. The field focuses on communication between computers and humans in natural language and NLP is all about making computers understand and generate human language.

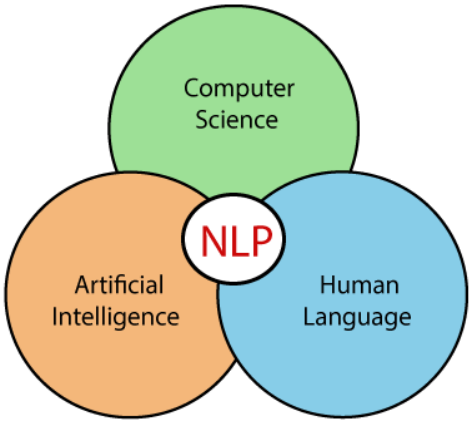


Fig 1.1 NLP Concepts

- The field blends computer science, linguistics and machine learning. The field is divided into the three parts:

- Speech recognition—the translation of spoken language into text.
- Natural language understanding—a computer's ability to understand language.
- Natural language generation—the generation of natural language by a computer.

Why NLP is needed?

- The essence of Natural Language Processing lies in making computers understand the natural language. That's not an easy task though. Computers can understand the structured form of data like spreadsheets and the tables in the database, but human languages, texts, and voices form an unstructured category of data, and it gets difficult for the computer to understand it, and there arises the need for Natural Language Processing.
- The primary goal of NLP is to enable computers to understand, interpret, and generate natural language, the way humans do.
- NLP techniques, including computational linguistics, machine learning, and statistical modeling are used to analyze, understand, and manipulate human language data, including text, speech, and other forms of communication.

History of NLP

(1940-1960)

- Focused on Machine Translation (MT)
- The Natural Languages Processing started in the year 1940s.
- 1948 - In the Year 1948, the first recognisable NLP application was introduced in Birkbeck College, London.
- 1950s - In the Year 1950s, there was a conflicting view between linguistics and computer science. Now, Chomsky developed his first book syntactic structures and claimed that language is generative in nature.
- In 1957, Chomsky also introduced the idea of Generative Grammar, which is rule based descriptions of syntactic structures.

(1960-1980)

- Flavored with Artificial Intelligence (AI)
- In the year 1960 to 1980, the key developments were:
- Augmented Transition Networks (ATN) - Augmented Transition Networks is a finite state machine that is capable of recognizing regular languages.
- Case Grammar was developed by Linguist Charles J. Fillmore in the year 1968. Case Grammar uses languages such as English to express the relationship between nouns and verbs by using the preposition.
- In the year 1960 to 1980, key systems were:

  - SHRDLU

    SHRDLU is a program written by Terry Winograd in 1968-70. It helps users to communicate with the computer and moving objects. It can handle instructions such as "pick up the green boll" and also answer the questions like "What is inside the black box." The main importance of SHRDLU is that it shows those syntax, semantics, and reasoning about the world that can be combined to produce a system that understands a natural language.

  - LUNAR

    LUNAR is the classic example of a Natural Language database interface system that is used ATNs and Woods' Procedural Semantics. It was capable of translating elaborate natural language expressions into database queries and handle 78% of requests without errors.

1980 – Current

- Till the year 1980, natural language processing systems were based on complex sets of hand-written rules. After 1980, NLP introduced machine learning algorithms for language processing.
- In the beginning of the year 1990s, NLP started growing faster and achieved good process accuracy, especially in English Grammar.
- In 1990 also, an electronic text introduced, which provided a good resource for training and examining natural language programs. Other factors may include the availability of computers with

fast CPUs and more memory. The major factor behind the advancement of natural language processing was the Internet.

- Now, modern NLP consists of various applications, like speech recognition, machine translation, and machine text reading.

How NLP Works?

NLP systems use machine learning algorithms to analyze large amounts of unstructured data and extract relevant information. The algorithms are trained to recognize patterns and make inferences based on those patterns. Here's how it works:

- The user must input a sentence into the Natural Language Processing (NLP) system.
- The NLP system then breaks down the sentence into smaller parts of words, called tokens, and converts audio to text.
- Then, the machine processes the text data and creates an audio file based on the processed data.
- The machine responds with an audio file based on processed text data.

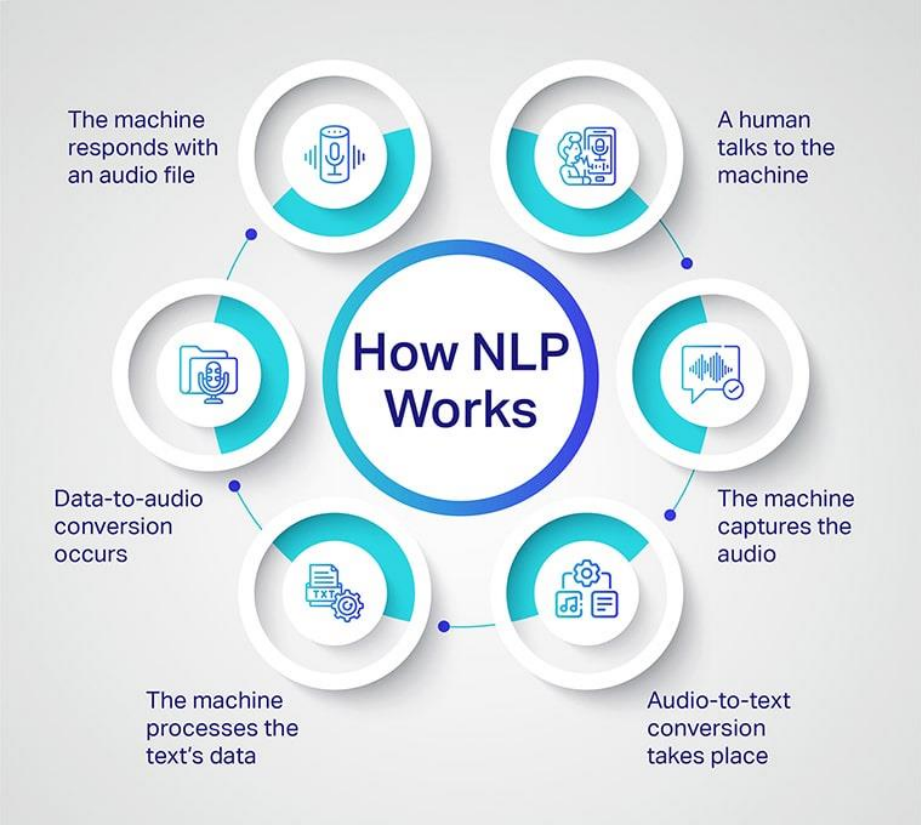


Fig 1.2 Working Principle of NLP

Applications of NLP

1. Text Classification: Classifying text into different categories based on their content, such as spam filtering, sentiment analysis, and topic modeling.
2. Named Entity Recognition (NER): Identifying and categorizing named entities in text, such as people, organizations, and locations.
3. Part-of-Speech (POS) Tagging: Assigning a part of speech to each word in a sentence, such as noun, verb, adjective, and adverb.
4. Sentiment Analysis: Analyzing the sentiment of a piece of text, such as positive, negative, or neutral.
5. Machine Translation: Translating text from one language to another.
6. Speech recognition and transcription: NLP techniques are used to convert speech to text, which is useful for tasks such as dictation and voice-controlled assistants.
7. Language translation: NLP techniques are used to translate text from one language to another, which is useful for tasks such as global communication and e-commerce.
8. Text summarization: NLP techniques are used to summarize long text documents into shorter versions, which is useful for tasks such as news summarization and document indexing.
9. Sentiment analysis: NLP techniques are used to determine the sentiment or emotion expressed in text, which is useful for tasks such as customer feedback analysis and social media monitoring.

Advantages of Natural Language Processing:

1. Improves human-computer interaction: NLP enables computers to understand and respond to human languages, which improves the overall user experience and makes it easier for people to interact with computers.
2. Automates repetitive tasks: NLP techniques can be used to automate repetitive tasks, such as text summarization, sentiment analysis, and language translation, which can save time and increase efficiency.
3. Enables new applications: NLP enables the development of new applications, such as virtual assistants, chatbots, and question answering systems, that can improve customer service, provide information, and more.
4. Improves decision-making: NLP techniques can be used to extract insights from large amounts of unstructured data, such as social media posts and customer feedback, which can improve decision-making in various industries.
5. Improves accessibility: NLP can be used to make technology more accessible, such as by providing text-to-speech and speech-to-text capabilities for people with disabilities.

6. Facilitates multilingual communication: NLP techniques can be used to translate and analyze text in different languages, which can facilitate communication between people who speak different languages.
7. Improves information retrieval: NLP can be used to extract information from large amounts of data, such as search engine results, to improve information retrieval and provide more relevant results.
8. Enables sentiment analysis: NLP techniques can be used to analyze the sentiment of text, such as social media posts and customer reviews, which can help businesses understand how customers feel about their products and services.
9. Improves content creation: NLP can be used to generate content, such as automated article writing, which can save time and resources for businesses and content creators.
10. Supports data analytics: NLP can be used to extract insights from text data, which can support data analytics and improve decision-making in various industries.
11. Enhances natural language understanding: NLP research and development can lead to improved natural language understanding, which can benefit various industries and applications.

Disadvantages of Natural Language Processing:

1. Limited understanding of context: NLP systems have a limited understanding of context, which can lead to misinterpretations or errors in the output.
2. Requires large amounts of data: NLP systems require large amounts of data to train and improve their performance, which can be expensive and time-consuming to collect.
3. Limited ability to understand idioms and sarcasm: NLP systems have a limited ability to understand idioms, sarcasm, and other forms of figurative language, which can lead to misinterpretations or errors in the output.
4. Limited ability to understand emotions: NLP systems have a limited ability to understand emotions and tone of voice, which can lead to misinterpretations or errors in the output.
5. Difficulty with multi-lingual processing: NLP systems may struggle to accurately process multiple languages, especially if they are vastly different in grammar or structure.
6. Dependency on language resources: NLP systems heavily rely on language resources, such as dictionaries and corpora, which may not always be available or accurate for certain languages or domains.
7. Difficulty with rare or ambiguous words: NLP systems may struggle to accurately process rare or ambiguous words, which can lead to errors in the output.

8. Lack of creativity: NLP systems are limited to processing and generating output based on patterns and rules, and may lack the creativity and spontaneity of human language use.
9. Ethical considerations: NLP systems may perpetuate biases and stereotypes, and there are ethical concerns around the use of NLP in areas such as surveillance and automated decision-making.

1.2 Challenges of NLP:

Fig 1.3 Challenges in NLP

- Misspellings

  Natural languages are full of misspellings, typos, and inconsistencies in style. For example, the word “process” can be spelled as either “process” or “processing.” The problem is compounded when you add accents or other characters that are not in your dictionary.

- Language Differences

  An English speaker might say, “I’m going to work tomorrow morning,” while an Italian speaker would say, “Domani Mattina vado al lavoro.” Even though these two sentences mean the same thing, NLP won’t understand the latter unless you translate it into English first.

- Words with Multiple Meanings

  NLP is based on the assumption that language is precise and unambiguous. In reality, language is neither precise nor unambiguous. Many words have multiple meanings and can be used in different ways. For example, when we say “bark,” it can either be dog bark or tree bark.

- Uncertainty and False Positives

  False positives occur when the NLP detects a term that should be understandable but can’t be replied to properly. The goal is to create an NLP system that can identify its limitations and clear up confusion by using questions or hints.

- Training Data

  One of the biggest challenges with natural processing language is inaccurate training data. The more training data you have, the better your results will be. If you give the system incorrect or biased data, it will either learn the wrong things or learn inefficiently.

NLP Examples:

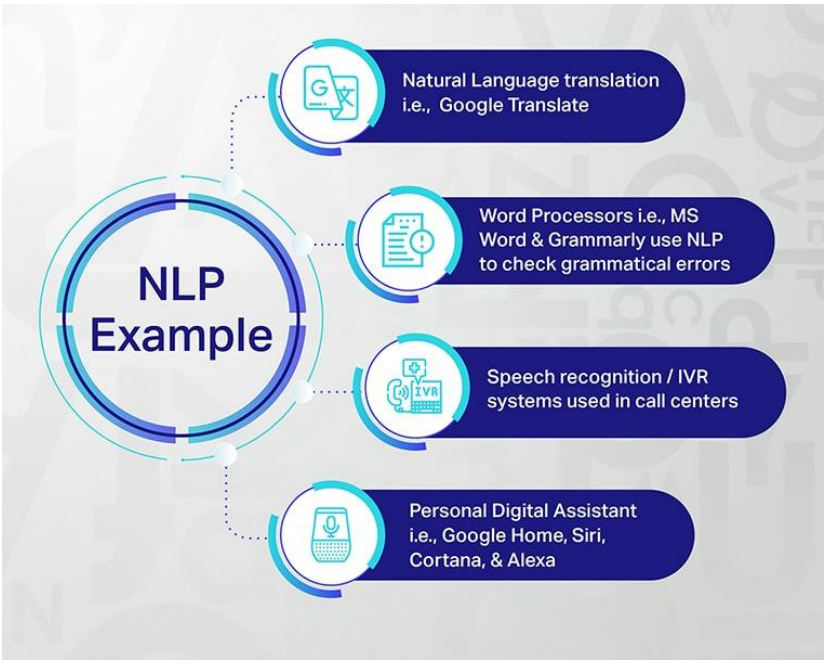


Fig 1.4 Examples for NLP

- Natural Language translation i.e., Google Translate

  Google Translate is a free web-based translation service that supports over 100 languages and can translate your content automatically into these languages. The service has two modes: translation and translation suggestions.

- Word Processors i.e., MS Word & Grammarly

  Word processors like MS Word and Grammarly use NLP to check text for grammatical errors. They do this by looking at the context of your sentence instead of just the words themselves.

- Speech recognition / IVR systems used in call centers

  Speech recognition is an excellent example of how NLP can be used to improve the customer experience. It is a very common requirement for businesses to have IVR systems in place so that customers can interact with their products and services without having to speak to a live person. This allows them to handle more calls but also helps cut costs.

- Personal Digital Assistants i.e., Google Home, Siri, Cortana, & Alexa

  The use of NLP has become more prevalent in recent years as technology has advanced. Personal Digital Assistant applications such as Google Home, Siri, Cortana, and Alexa have all been updated with NLP capabilities. These devices use NLP to understand human speech and respond appropriately.

1.3 Phases in NLP:

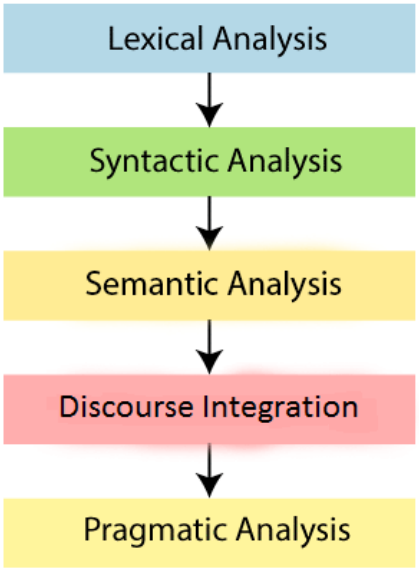


Fig 1.5 Phases in NLP

1. Lexical Analysis and Morphological

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. It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in that particular language. The lexical analysis divides the text into paragraphs, sentences, and words. So we need to perform Lexicon Normalization.

The most common lexicon normalization techniques are Stemming:

- Stemming: Stemming is the process of reducing derived words to their word stem, base, or root form—generally a written word form like-“ing”, “ly”, “es”, “s”, etc.
- Lemmatization: Lemmatization is the process of reducing a group of words into their lemma or dictionary form. It takes into account things like POS(Parts of Speech), the meaning of the word in the sentence, the meaning of the word in the nearby sentences, etc. before reducing the word to its lemma.

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Chennai goes to the John

In the real world, Chennai goes to the John, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

Syntactical parsing involves the analysis of words in the sentence for grammar. Dependency Grammar and Part of Speech (POS)tags are the important attributes of text syntactic.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences. It retrieves the possible meanings of a sentence that is clear and semantically correct. Its process of retrieving meaningful insights from text.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it. It is nothing but a sense of context. That is sentence or word depends upon that sentences or words. It's like the use of proper nouns/pronouns.

For example, Ram wants it.

In the above statement, we can clearly see that the "it" keyword does not make any sense. In fact, it is referring to anything that we don't know. That is nothing but this "it" word depends upon the previous sentence which is not given. So once we get to know about "it", we can easily find out the reference.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues. It means the study of meanings in a given language. Process of extraction of insights from the text. It includes the repetition of words, who said to whom? etc. It understands that how people communicate with each other, in which context they are talking and so many aspects.

For Example: "Open the door" is interpreted as a request instead of an order.

1.4 Outline of English syntax:

For NLP analysis, there are four aspects of syntax that are most important: the syntactic categories and features of individual words, which we also call their parts of speech; the well-formed sequences of words into phrases and sentences, which we call constituency; the requirements that some words have for other co-occurring constituents, which we call subcategorization; and binary relations between words that are the lexical heads (main word) of a constituent, which we call lexical dependency, or just "dependency".

Word Types and Features

In English, we typically think of the word as the smallest unit. However, trained linguists make some finer grained distinctions. For example, linguists use the term lemma, or root, or base form to describe the canonical form of a word that has several variations for forming the plural or a particular tense.

- For common nouns, like "apple", this would be the singular form.

- For verbs, it is the untensed form (that is the one that would follow the word "to" in an infinitive, such as "to eat", "to be", or "to go").

- Linguists use the term lexeme to describe a word type – which includes the set of the lemma and all its variants.
- The term morpheme is used to describe strings that carry meaning but may be smaller than a word, such as prefixes (which are substrings at the front of a word) and suffixes (which are substrings at the end of a word). Both can add either syntactic or semantic information to a word.
- Analyzing a word into morphemes is called "morphology".
- Finding the root is called lemmatization.
- NLP work sometimes uses the notion of "stems" instead of roots. Stems are substrings of a word, which can depend on an implementation, as there is no standard form. They are useful for specifying patterns to match all members of a lexeme.
- NLP work also uses the term "token", which is an instance of a word as it occurs in use. So, if a sentence includes the same word twice, there will be two separate tokens created for it.

Broad syntactic categories of words and the syntactic attributes that occur as variants of spelling are categorized in ten types: nouns, pronouns, proper nouns, determiners, verbs, prepositions, adverbs, adjectives, conjunctions, and wh-words.

Nouns, Pronouns, and Proper Nouns

Nouns are used to name or describe entities, which might be physical (such as "cat" or "rock") or abstract (such as "freedom" or "laughter") or both (such as "city" or "company"). Nouns can be singular or plural. The plural form is usually marked with the suffix "s" or "es", as in "cats"; if a word ends in "y", it is changed to "i" before adding the suffix. i.e, Regular plural form.

Example of nouns with a plural formed with -s or -es suffix

Singular form	Regular plural form
frog	frogs
idea	ideas
fly	flies

fox	foxes
class	classes

Some plurals are irregular, as in “children” or “knives”. Some nouns (called “count” nouns), unless they are plural, require a determiner or cardinal number to specify the denoted set, e.g. “the boy” or “three boys”.

Example of nouns with an irregular plural

Singular form	Irregular plural form
child	children
sheep	sheep
goose	geese
knife	knives

Nouns occur in the subjects of sentences and as objects following a verb or preposition.

Typical placement of nouns

Noun in the subject	Main verb	Noun in an object	Noun in a prepositional phrase
The boy	put	his towel	in his locker

Proper nouns are the names of people, places, and things and are capitalized wherever they occur, as in “My name is Susan”. Proper nouns rarely appear as plurals, but since they sometimes do, as in “We visited the Smiths”, NLP systems include a category for plural proper nouns.

Pronouns are used to refer to people and things that have been mentioned before or presupposed to exist. They have different forms to specify whether they are singular or plural and their syntactic role (subject or object). In a grammatical sentence, the form should agree with the properties of the verb, although current NLP systems often ignore these features and only use only one category. One subclass of pronouns that is distinguished are those that express possession, and can be used in place of a determiner, e.g., “my book”

or “your house,” and this subclass may also be assigned a separate part of speech. Also, some pronouns are used to form a question and thus also merit their own labels. They include both regular wh-pronouns, including “what”, “who”, and “whom,” and possessive wh-pronouns, such as “whose”.

Common nouns and proper nouns are considered an open class of words, which means people may invent new ones to describe new objects or names. By contrast, pronouns are considered a closed class of words. With open-class words, algorithms must address that new items might occur that will be outside of the known vocabulary.

Determiners

Determiners include “the”, “a”, “an”, “that”, “these”, “this”, and “those”. There are also determiners that are used in questions, such as “what” and “which”. Determiners only occur in noun phrases, before any adjectives or nouns. Some common nouns, when they express a mass quantity, like “water” or “rice”, or when they are plural like “cats”, do not require a determiner. Proper nouns generally do not allow a determiner, except when they are plural, e.g., “The Smiths” or when it is part of the name itself, e.g., “The Ohio State University”. Possessive phrases, which are marked with an apostrophe and the suffix “-s”, can take the place of a determiner, as in the phrase “my mother’s house”. Pronouns, regular or possessive, are *never* preceded by determiners. Determiners are considered a closed class of words.

Verbs and Auxiliary Verbs

Verbs are usually tensed (past, present, future). They include both verbs where the tensed forms are regular or irregular. Also, in some contexts, verbs can appear untensed, such as after an auxiliary or after the word “to”. Verbs are also marked for number (singular or plural), and for person. First person is “I”; second person is “you”; and third person is “he”, “she”, or “it”. The third-person singular form is marked with “-s”; the non-3rd person singular present looks the same as the root form. Verbs also have participle forms for past (eg., “broken” or “thought”) and present (e.g., “thinking”).

Some verbs require a particle which is similar to a preposition except that it forms an essential part of the meaning of the verb that can be moved either before or after another argument, as in “she took off her hat” or “she took her hat off”.

Verbs that can be main verbs are an open class. Verbs that are modals or auxiliary verbs (also called helping verbs) are a closed class. They are used along with a main verb to express ability ("can", "could"), possibility ("may", "might"), necessity ("shall", "should", "ought"), certainty ("do", "did"), future ("will", "would"), past ("has", "had", "have", "was", "were"). NLP systems treat modals and auxiliaries as a separate part of speech. They are also all irregular in the forms that they take for different combinations of features, such as past, plural, etc. For example, the modal "can" uses the form "can" for any value for number and "could" for any value for "past".

Example of a regular verb and some suffixes

Example	Regular Forms	Suffix	Features
walk	walks; walked; walking	-s; -ed; -ing	3rd person singular, present; past; participle

Some irregular verb forms

Example	Irregular forms	Features
break	broke; broken	past; past participle
eat	ate; eaten	past; past participle
sit	sat; seated	past; past participle

Prepositions

Prepositions, such as "with", "of", "for", and "from" are words that relate two nouns or a noun and a verb. Prepositions require a noun phrase argument (to form a prepositional phrase). It is estimated that there about 150 different prepositions (including 94 one-word prepositions and 56 complex prepositions, such as "out of")[14]. Prepositions are generally considered a closed class, but the possibility of complex combinations suggests that algorithms might be better off allowing for out of vocabulary examples.

Adjectives and Adverbs

Adjectives normally modify nouns, as in "the big red book", but may also be an argument of a verb (including forms of "be", "feel", "appear", and "become"). Adjectives can also be marked as comparative (meaning "more than typical", using the suffix "-er") or superlative (meaning "more than any others", using

the suffix "-est"). Adverbs modify verbs, adjectives, or other adverbs. They express manner or intensity. They may be comparative (e.g., "better") or superlative (e.g., "best"). Adverbs that end in the suffix "-ly" have been derived from a related adjective (e.g., "quickly "is derived from "quick ").

Conjunctions

Conjunctions, such as "and", "although", "because", "but", "however", "or", "nor", "so", "unless", "when", "where", "while", etc. are words that join words, phrases, clauses, or sentences. They can be discontinuous, e.g,. "either … or", "neither … nor", "both … and", "not only … but also", "on the one hand … on the other (hand)", "not just … but", and "not only … but".

Wh-Words

Wh-words that begin with the letters "wh-" like "who", "what", "when", "where", "which", "whose", and "why" and their close cousins "how", "how much", "how many", etc. They are used for posing questions and are thus sometimes called interrogatives. Unlike the word types mentioned so far, they can be determiners, adverbs, or pronouns (both regular and possessive), and so it is typical to see them marked as a special subtype of each. Identifying phrases that include wh-words is important, because they usually occur near the front in written text and fill an argument role that has been left empty in its normal position, as in "Which book did you like best?" In informal speech one might say "You left your book where?" or "You said what?", but the unusual syntax also suggests a problem (like mishearing, shock, or criticism). The semantics of the wh-expression specify what sort of answer the speaker is expecting (e.g., a person, a description, a time, a place, etc) and thus are essential to question-answering systems.

1.4.1 Text planning

It refers to the process of organizing and structuring the content of a piece of writing before actually drafting the text. It involves thinking about the overall purpose of the text, identifying key ideas, and determining the most effective way to present information. Proper text planning can improve the clarity and coherence of your writing. Here are some steps you can take in the text planning phase:

1. **Define Purpose and Audience:**
    - Understand the purpose of your text. Are you informing, persuading, entertaining, or explaining?
    - Identify your target audience and tailor your message accordingly.
2. **Generate Ideas:**
    - Brainstorm and jot down all relevant ideas and information related to your topic.
    - Prioritize these ideas based on importance and relevance.
3. **Create an Outline:**
    - Develop a clear structure for your text. This could include an introduction, body paragraphs, and a conclusion.
    - Organize your main ideas and supporting details logically.
4. **Thesis Statement or Main Idea:**
    - Clearly state the main point or thesis of your text. This provides focus and direction for your writing.
5. **Logical Flow:**
    - Ensure a smooth and logical flow between paragraphs and sections. Use transitional phrases to guide the reader from one point to the next.
6. **Consider Tone and Style:**
    - Determine the appropriate tone for your audience and purpose. Are you writing formally, informally, or somewhere in between?
    - Choose a writing style that matches the nature of your content.
7. **Review and Revise:**
    - Take a critical look at your plan and assess whether it effectively conveys your intended message.
    - Make revisions as needed to improve clarity and coherence.
8. **Check for Consistency:**
    - Ensure consistency in terms of terminology, formatting, and style throughout your text.
9. **Include Supporting Evidence:**
    - If applicable, plan for the inclusion of evidence, examples, or data to support your points.

10. **Consider Length and Structure:**
    - Determine the appropriate length for your text and adjust the level of detail accordingly.
    - Ensure that the structure of your text suits the nature of your content (e.g., chronological, cause and effect, problem-solution).

1.5 Grammar and Parsing in NLP

Grammars

Grammar is defined as the rules for forming well-structured sentences. Grammar also plays an essential role in describing the syntactic structure of well-formed programs, like denoting the syntactical rules used for conversation in natural languages.

- In the theory of formal languages, grammar is also applicable in Computer Science, mainly in programming languages and data structures. Example - In the C programming language, the precise grammar rules state how functions are made with the help of lists and statements.
- Mathematically, a grammar G can be written as a 4-tuple (N, T, S, P) where:
    - **N or VN** = 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 $(V_N \cup \sum)$, and at least one symbol of α belongs to VN

Syntax

Each natural language has an underlying structure usually referred to under Syntax. The fundamental idea of syntax is that words group together to form the constituents like groups of words or phrases which behave as a single unit. These constituents can combine to form bigger constituents and, eventually, sentences.

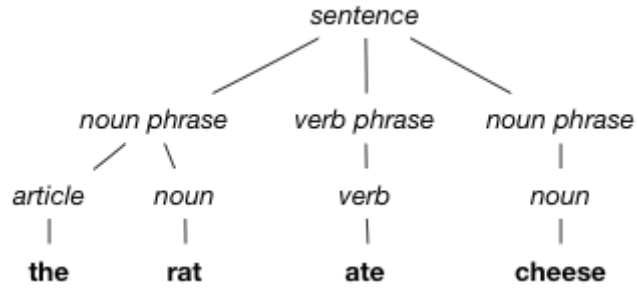
- Syntax describes the regularity and productivity of a language making explicit the structure of sentences, and the goal of syntactic analysis or parsing is to detect if a sentence is correct and provide a syntactic structure of a sentence.

Syntax also refers to the way words are arranged together. Let us see some basic ideas related to syntax:

- **Constituency:** Groups of words may behave as a single unit or phrase - A constituent, for example, like a Noun phrase.
- **Grammatical relations:** These are the formalization of ideas from traditional grammar. Examples include - subjects and objects.
- **Subcategorization and dependency relations:** These are the relations between words and phrases, for example, a Verb followed by an infinitive verb.
- **Regular languages and part of speech:** Refers to the way words are arranged together but cannot support easily. Examples are Constituency, Grammatical relations, and Subcategorization and dependency relations.
- **Syntactic categories and their common denotations in NLP:** np - noun phrase, vp - verb phrase, s - sentence, det - determiner (article), n - noun, tv - transitive verb (takes an object), iv - intransitive verb, prep - preposition, pp - prepositional phrase, adj – adjective

How grammar is used to create parse tree:

- Parsing is the process of converting a stream of input into a structured representation. The input stream may consists of words, characters, or even bits. The output of the process is a tree.
- Our brains are remarkably good at parsing. When we hear a sentence like "The rat ate cheese," our brains build a **parse tree** similar to the following diagram:
-



-

A parser in NLP uses the grammar rules (formal grammar rules) to verify if the input text is valid or not syntactically. The parser helps us to get the meaning of the provided text (like the dictionary meaning of the provided text).

A parser uses the grammar rules (formal grammar rules) to verify if the input text is valid or not syntactically. So, what is grammar? Well, grammar is a set of rules developed to verify the words of a language. Grammar helps us to form a well-structured sentence.

Parsing plays a vital role in the NLP process.

1. Sentiment Analysis.
2. Relation Extraction.
3. Question Answering.
4. Speech Recognition.
5. Machine Translation.
6. Grammar Checking.

Top Down Parsing

The top-down parsing approach follows the left-most derivation technique. In the top-down parsing approach, the construction of the parse tree starts from the root node. So, first, the root node is formed and then the generation goes subsequently down by generating the leaf node (following the top to down approach).

- The top-down parsing uses the leftmost derivation approach to construct the parsing tree of the text.
- The top-down parser in NLP does not support the grammar having common prefixes.
- The top-down parser in NLP also guarantees that the grammar is free from all ambiguity and has followed the left recursion.

We can perform the top-down parsing using two ways:

1. Using backtracking.
2. Without using backtracking.

The recursive descent parser in NLP follows the top-down parsing approach. This parser checks the syntax of the input stream of text by reading it from left to right (hence, it is also known as the Left-Right Parser). The parser first reads a character from the input stream and then verifies it or matches it with the grammar's terminals. If the character is verified, it is accepted else it gets rejected.

Bottom-up Parsing

The bottom-up parsing approach follows the leaves-to-root approach or technique. In the bottom-up parsing approach, the construction of the parse tree starts from the leaf node. So, first, the leaf node is formed and then the generation goes subsequently up by generating the parent node, and finally, the root node is generated (following the bottom-to-up approach). The main aim of bottom-up parsing is to reduce the input string of text to get the start symbol by using the rightmost derivation technique.

- he bottom-up parsing approach is used by various programming language compilers such as C++, Perl, etc.
- This technique can be implemented very efficiently.
- The bottom-up parsing technique detects the syntactic error faster than other parsing techniques.

The bottom-up parsers or LR parsers are of four types:

1. LR(0)
2. SLR(1) or Simple LR
3. LALR or LookAhead LR
4. CLR or Canonical LR

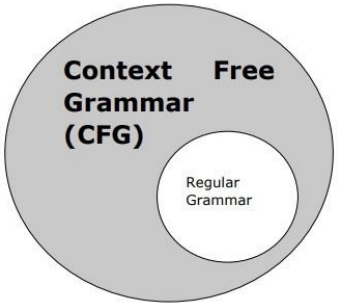
Here, the term or symbol L means that we are reading the text from left to right and the symbol R means that we are using the rightmost derivation technique to generate the parse tree.

Different Types of Grammar in NLP

1. Context-Free Grammar (CFG)
2. Constituency Grammar (CG)
3. Dependency Grammar (DG)

Context Free Grammar

Context-free grammar consists of a set of rules expressing how symbols of the language can be grouped and ordered together and a lexicon of words and symbols.



- One example rule is to express an NP (or noun phrase) that can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal, a Nominal in turn can consist of one or more Nouns: NP → DetNominal, NP → ProperNoun; Nominal → Noun | NominalNoun
- Context-free rules can also be hierarchically embedded, so we can combine the previous rules with others, like the following, that express facts about the lexicon: Det → a Det → the Noun → flight
- Context-free grammar is a formalism power enough to represent complex relations and can be efficiently implemented. Context-free grammar is integrated into many language applications
- A Context free grammar consists of a set of rules or productions, each expressing the ways the symbols of the language can be grouped, and a lexicon of words

Context-free grammar (CFG) can also be seen as the list of rules that define the set of all well-formed sentences in a language. Each rule has a left-hand side that identifies a syntactic category and a right-hand side that defines its alternative parts reading from left to right. - **Example:** The rule s --> np vp means that "a sentence is defined as a noun phrase followed by a verb phrase."

- **Formalism in rules for context-free grammar:** A sentence in the language defined by a CFG is a series of words that can be derived by systematically applying the rules, beginning with a rule that has s on its left-hand side.
    - Use of parse tree in context-free grammar: A convenient way to describe a parse is to show its parse tree, simply a graphical display of the parse.
    - A parse of the sentence is a series of rule applications in which a syntactic category is replaced by the right-hand side of a rule that has that category on its left-hand side, and the final rule application yields the sentence itself.
- **Example:** the rule s --> np vp means that "a sentence is defined as a noun phrase followed by a verb phrase." Figure 1 shows a simple CFG that describes the sentences from a small subset of English

Figure 1. A grammar and a parse tree for "the giraffe dreams".

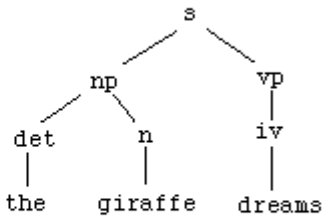
```
s   → np vp

np  → det n
vp  → tv np
    → iv

det → the
    → a
    → an

n   → giraffe
    → apple

iv  → dreams
tv  → eats
    → dreams
```



- A *sentence* in the language defined by a CFG is a series of words that can be derived by systematically applying the rules, beginning with a rule that has s on its left-hand side.
- A *parse* of the sentence is a series of rule applications in which a syntactic category is replaced by the right-hand side of a rule that has that category on its left-hand side, and the final rule application yields the sentence itself. E.g., a parse of the sentence "the giraffe dreams" is:

s => np vp => det n vp => the n vp => the giraffe vp => the giraffe iv => the giraffe dreams

- A convenient way to describe a parse is to show its *parse tree*, which is simply a graphical display of the parse.

Classification of Symbols in CFG

- The symbols that correspond to words in the language, for example, the nightclub, are called terminal symbols, and the lexicon is the set of rules that introduce these terminal symbols.
- The symbols that express abstractions over these terminals are called non-terminals.
- In each context-free rule, the item to the right of the arrow (→) is an ordered list of one or more terminals and non-terminals, and to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. - The non-terminal associated with each word in the lexicon is its lexical category or part of speech.

A context-free grammar G is defined by four parameters N, Σ, P, S (technically “is a 4-tuple”):

- N a set of non-terminal symbols (or variables)
- Σ a set of terminal symbols (disjoint from N)

- R a set of rules or productions, each of the form $A \rightarrow \beta$, where A is a non terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
- S a designated start symbol

Set of Non-terminals

It is represented by V. The non-terminals are syntactic variables that denote the sets of strings, which helps in defining the language that is generated with the help of grammar.

Set of Terminals

It is also known as tokens and represented by Σ. Strings are formed with the help of the basic symbols of terminals.

Set of Productions

It is represented by P. The set gives an idea about how the terminals and nonterminals can be combined. Every production consists of the following components:

- Non-terminals,
- Arrow,
- Terminals (the sequence of terminals).
- The left side of production is called non-terminals while the right side of production is called terminals.

Start Symbol

The production begins from the start symbol. It is represented by symbol S. Non-terminal symbols are always designated as start symbols.

Limitations of context-free grammar

- **Limited expressiveness:** Context-free grammar is a limited formalism that cannot capture certain linguistic phenomena such as idiomatic expressions, coordination and ellipsis, and even long-distance dependencies.
- **Handling idiomatic expressions:** CFG may also have a hard time handling idiomatic expressions or idioms, phrases whose meaning cannot be inferred from the meanings of the individual words that make up the phrase.

- **Handling coordination:** CFG needs help to handle coordination, which is linking phrases or clauses with a conjunction.
- **Handling ellipsis**: Context-free grammar may need help to handle ellipsis, which is the omission of one or more words from a sentence that is recoverable from the context.

The limitations of context-free grammar can be mitigated by using other formalisms such as dependency grammar which is powerful but more complex to implement, or using a hybrid approach where both constituency and dependency are used together. We can also additionally use machine learning techniques in certain cases.

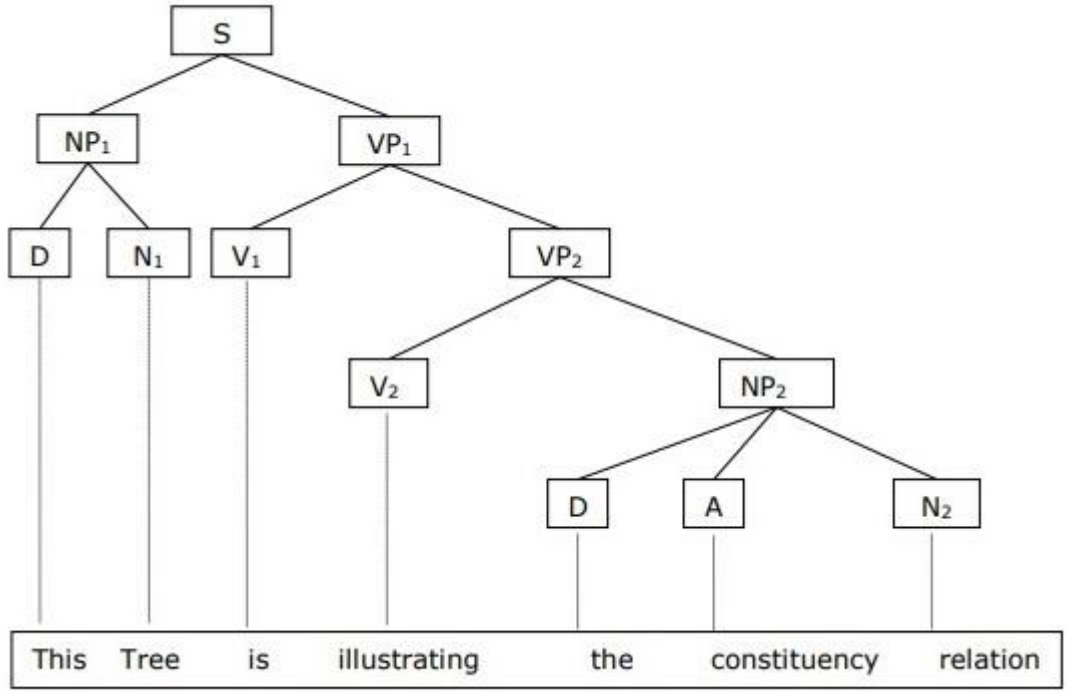
Constituency Grammar

Constituency Grammar is also known as Phrase structure grammar. Furthermore, it is called constituency Grammar as it is based on the constituency relation. It is the opposite of dependency grammar.

- The constituents can be any word, group of words or phrases in Constituency Grammar. The goal of constituency grammar is to organize any sentence into its constituents using their properties.
- Characteristic properties of constituency grammar and constituency relation:
    - All the related frameworks view the sentence structure in terms of constituency relation.
    - To derive the constituency relation, we take the help of subject-predicate division of Latin as well as Greek grammar.
    - In constituency grammar, we study the clause structure in terms of noun phrase NP and verb phrase VP.
- The properties are derived generally with the help of other NLP concepts like part of speech tagging, a noun or Verb phrase identification, etc. For example, Constituency grammar can organize any sentence into its three constituents - a subject, a context, and an object.

For Example,

Sentence: This tree is illustrating the constituency relation



In Constituency Grammar, the constituents can be any word, group of words, or phrases and the goal of constituency grammar is to organize any sentence into its constituents using their properties.

For Example, constituency grammar can organize any sentence into its three constituents- a subject, a context, and an object.
Sentence: <subject> <context> <object>

These three constituents can take different values and as a result, they can generate different sentences. For Example, If we have the following constituents, then
<subject> The horses / The dogs / They
<context> are running / are barking / are eating
<object> in the park / happily / since the morning

Example sentences that we can be generated with the help of the above constituents are:
“The dogs are barking in the park”
“They are eating happily”

"The horses are running since the morning"

Limitations of Constituency grammar

- Constituency grammar is not language-specific, making it easy to use the same model for multiple languages or switch between languages, hence handling the multilingual issue plaguing the other two types of grammar.
- Since constituency grammar uses a parse tree to represent the hierarchical relationship between the constituents of a sentence, it can be easily understood by humans and is more intuitive than other representation grammars.
- Constituency grammar also is simple and easier to implement than other formalisms, such as dependency grammar, making it more accessible for researchers and practitioners.
- Constituency grammar is robust to errors and can handle noisy or incomplete data.
- Constituency grammar is also better equipped to handle coordination which is the linking of phrases or clauses with a conjunction.

Dependency Grammar

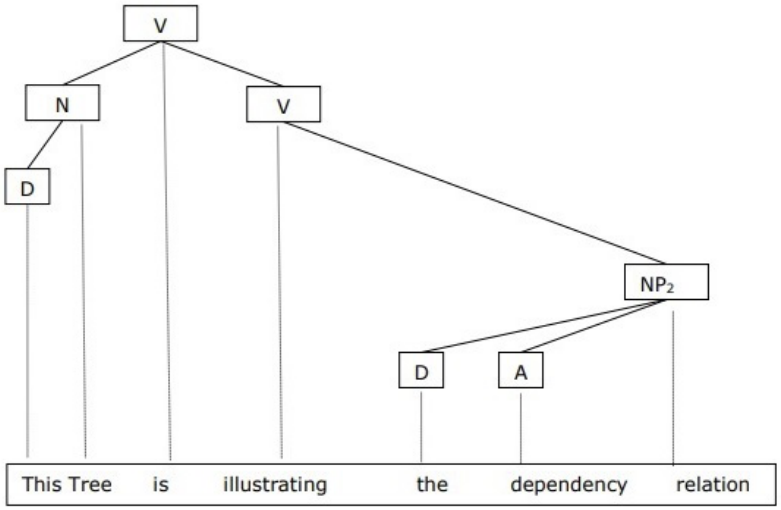
Dependency Grammar is the opposite of constituency grammar and is based on the dependency relation. It is opposite to the constituency grammar because it lacks phrasal nodes.

Dependency grammar and dependency relation:

- Dependency Grammar states that words of a sentence are dependent upon other words of the sentence. These Words are connected by directed links in dependency grammar. The verb is considered the center of the clause structure.
- Dependency Grammar organizes the words of a sentence according to their dependencies. Every other syntactic unit is connected to the verb in terms of a directed link. These syntactic units are called dependencies.
    - One of the words in a sentence behaves as a root, and all the other words except that word itself are linked directly or indirectly with the root using their dependencies.
    - These dependencies represent relationships among the words in a sentence, and dependency grammar is used to infer the structure and semantic dependencies between the words.
- It is opposite to the constituency grammar and is based on the dependency relation.

- Dependency grammar (DG) is opposite to constituency grammar because it lacks phrasal nodes. In Dependency Grammar, the words are connected to each other by directed links.
- The verb is considered the center of the clause structure.
- Every other syntactic unit is connected to the verb in terms of directed link. These syntactic units are called dependencies.

For Example,
Sentence: This tree is illustrating the dependency relation



Dependency Grammar states that words of a sentence are dependent upon other words of the sentence.

Limitations of Dependency grammar

Ambiguity: Dependency grammar has issues with ambiguity when it comes to interpreting the grammatical relationships between words, which are particularly challenging when dealing with languages that have rich inflections or complex word order variations.

- **Data annotation:** Dependency parsing also requires labeled data to train the model, which is time-consuming and difficult to obtain.

- **Handling long-distance dependencies:** Dependency parsing also has issues with handling long-term dependencies in some cases where the relationships between words in a sentence may be very far apart, making it difficult to accurately capture the grammatical structure of the sentence.
- **Handling ellipsis and coordination:** Dependency grammar also has a hard time handling phenomena that are not captured by the direct relationships between words, such as ellipsis and coordination, which are typically captured by constituency grammar.

The limitations of dependency grammar can be mitigated by using constituency grammar which, although less powerful, but more intuitive and easier to implement. We can also use a hybrid approach where both constituency and dependency are used together, and it can be beneficial.

- Comparing Constituency grammar with Dependency grammar: Constituency grammar focuses on grouping words into phrases and clauses, while dependency grammar focuses on the relationships between individual words. Each word in a sentence is represented by a node in a dependency graph, and the edges between nodes represent the grammatical relationships between the words.
    - Dependency grammar is typically more powerful for some languages and NLP tasks as it captures the relationships between the words more accurately, but also more complex to implement and less intuitive.
    - Constituency grammar is more intuitive and easier to implement but can be less expressive.

1.5 Features of Augmented Grammars

An augmented grammar is any grammar whose productions are augmented with **conditions** expressed using features. Features may be associated with any nonterminal symbol in a derivation. A feature associated with a nonterminal symbol is shown following that nonterminal separated from it by a ".", e.g. *A.COUNT* is the *COUNT* feature associated with the nonterminal *A*. When the value is included with the feature, the feature and its value are bracketed and the dot omitted, as in *A*[*COUNT 1*]. If no ambiguity arises, the feature name may be omitted, as in *A*[*1*]. The values of the features may be **assigned** by the application of a production (indicated by the assignment operator ":=" for 'is set equal to'), or they may be **compared** (**checked**) using a comparison operator such as "=".

Examples

An augmented context-free grammar for generating the strictly indexed language $L_{n^3} = \{a^n b^n c^n : n > 0\}$

Here are the productions of an augmented context-free grammar G_1 that **generates** the SIXL $L_{n^3} = \{a^n b^n c^n \mid n > 0\}$. Nonterminal symbols that occur more than once in a production are distinguished by

indices. The nonterminal symbol to the left of the rewrite arrow "® " gets index 0 (e.g. A_0), and recurring nonterminals to the right of the arrow get indices 1, 2, ... from left to right, e.g. A_1. *COUNT* is an **integer feature** (a feature that takes integer values) of each of the nonterminal symbols in G_1.

- S ® A B C, where A.COUNT := S.COUNT, B.COUNT := S.COUNT, C.COUNT := S.COUNT
- A_0 ® a A_1, where A_1.COUNT := A_0.COUNT - 1
- B_0 ® b B_1, where B_1.COUNT := B_0.COUNT - 1
- C_0 ® c C_1, where C_1.COUNT := C_0.COUNT - 1
- A ® e, where A.COUNT = 0
- B ® e, where B.COUNT = 0
- C ® e, where C.COUNT = 0

An augmented context-free grammar for parsing L_{n^3}

Here are the productions of a grammar G_2 that **parses** L_{n^3}. A derivation with respect to a **parsing grammar** (**parser**) such as G_2, is constructed in reverse fashion from that of a **generating grammar** (**generator**) such as G_1. [Note that I do not call G_2 a *generative* grammar. I use the latter term, as Chomsky does, to refer to the class of parsers and generators of languages.] It starts with a terminal string, and each successive line is built up from the preceding line by replacing a substring by a nonterminal symbol together with its features in accordance with a production of the grammar. The derivation is terminated if the final line is a start symbol, and it's easy to see that the language parsed by G_2 is identical to the language generated by G_1, namely L_{n^3}. Each equivalence class of derivations with respect to a parser can be represented as a tree. However, the structural descriptions associated with the members of L_{n^3} by G_2 are different from those associated with them by G_1. To distinguish the productions of a parser from those of a generator, I also reverse the direction of the production arrow.

- S ¬ A B C, where A.COUNT = B.COUNT = C.COUNT
- A_0 ¬ A_1 A_2, where A_0.COUNT := A_1.COUNT + A_2.COUNT
- B_0 ¬ B_1 B_2, where B_0.COUNT := B_1.COUNT + B_2.COUNT
- C_0 ¬ C_1 C_2, where C_0.COUNT := C_1.COUNT + C_2.COUNT
- A ¬ a, where A.COUNT := 1
- B ¬ b, where B.COUNT := 1
- C ¬ c, where C.COUNT := 1

Write the productions of a parser that is structurally equivalent to the generator G_1

The productions of the parser that is structurally equivalent to the generator G_1 are as follows.

- S ¬ A B C, where A.COUNT = B.COUNT = C.COUNT
- A_0 ¬ a A_1, where A_0.COUNT := A_1.COUNT + 1
- B_0 ¬ b B_1, where B_0.COUNT := B_1.COUNT + 1
- C_0 ¬ c C_1, where C_0.COUNT := C_1.COUNT + 1
- A ¬ e, where A.COUNT := 0
- B ¬ e, where B.COUNT := 0
- C ¬ e, where C.COUNT := 0

The generator G_1 makes use of an infinite set of start symbols $\{S[n]: n > 0\}$. The generator G_3, whose productions are listed below, makes use of the single start symbol $A[0]$. Note that G_3 is an augmented regular grammar. *NEWCOUNT* , like *COUNT* (but defined for nonterminal *B* only), is an integer feature.

- A_0 ® a A_1, where A_1.COUNT := A_0.COUNT + 1
- A ® b B, where B.COUNT := A.COUNT, B.NEWCOUNT := A.COUNT - 1
- B_0 ® b B_1, where B_1.COUNT := B_0.COUNT, B_1.NEWCOUNT := B_0.NEWCOUNT - 1
- B ® c C, where B.NEWCOUNT = 0, C.COUNT := B.COUNT - 1
- C_0 ® c C_1, where C_1.COUNT := C_0.COUNT - 1
- C ® e, where C.COUNT = 0

Write the productions of a parser that is structurally equivalent to the generator G_1

Here are the productions of an augmented right-to-left regular parser for L_{n^3}. *The start symbol is* A*[0]*.

- A_0 ¬ a A_1, where A_0.COUNT := A_1.COUNT - 1
- A ¬ a B, where A.COUNT := B.COUNT-1, B.NEWCOUNT = 0
- B_0 ¬ b B_1, where B_0.COUNT := B_1.COUNT, B_0.NEWCOUNT := B_1.NEWCOUNT - 1
- B ¬ b C, where B.COUNT := C.COUNT, B.NEWCOUNT := C.COUNT - 1
- C_0 ¬ c C_1, where C_0.COUNT := C_1.COUNT + 1
- C ¬ c, where C.COUNT := 1

Here are the productions of an augmented left-to-right regular parser for L_{n^3}. *The start symbol is* C*[0]*.

- C_0 ¬ C_1 c, where C_0.COUNT := C_1.COUNT - 1
- C ¬ B c, where C.COUNT := B.COUNT - 1, B.NEWCOUNT = 0

- $B_0 \neg B_1$ b, where B_0.COUNT := B_1.COUNT, B_0.NEWCOUNT := B_1.NEWCOUNT - 1
- B ¬ A b, where B.COUNT := A.COUNT, B.NEWCOUNT := A.COUNT - 1
- $A_0 \neg A_1$ a, where A_0.COUNT := A_1.COUNT + 1
- A ¬ a, where A.COUNT := 1

1.5.1 Features in linguistic analysis

Linguistic analysis is the theory behind what the computer is doing. We say that the computer is performing Natural Language Processing (NLP) when it is doing an analysis based on the theory. Linguistic analysis is the basis for Text Analytics.

Features play a crucial role in linguistic analysis as they provide valuable information for understanding the structure and meaning of natural language. Features help disambiguate meanings, refine interpretations, and contribute to the overall precision of language understanding. Here, we will explore the importance of features like gender, number, and tense in linguistic analysis.

1. Gender

   - In many languages, gender is an important grammatical feature. For example, in languages like Spanish or French, nouns are assigned either a masculine or feminine gender. Understanding the gender of nouns is crucial for proper agreement with articles, adjectives, and pronouns.
   - Example: In Spanish, "el libro" (the book) uses the masculine article "el" because "libro" (book) is a masculine noun. Similarly, "la mesa" (the table) uses the feminine article "la" because "mesa" (table) is a feminine noun.

2. Number

   - Number refers to the grammatical category that indicates whether a noun or pronoun is singular or plural. Agreement in number is essential for maintaining grammatical correctness in a sentence.
   - Example: In English, "cat" is singular, and "cats" is plural. The verb form also changes accordingly, e.g., "The cat is sleeping" vs. "The cats are sleeping."

3. Tense

   - Tense is a temporal feature that indicates the time of an action or state. Different tenses convey when an action occurred or will occur, providing essential context for understanding the meaning of a sentence.
   - Example: In English, "I walk" (present tense), "I walked" (past tense), and "I will walk" (future tense) convey different temporal aspects of the action.

4. Person

    - Person indicates the relationship between the speaker, the listener, and the entities being discussed. It helps in determining who is performing the action or being referred to in a sentence.
    - Example: In English, "I walk," "you walk," and "he/she/it walks" demonstrate the first, second, and third person respectively.

5. Aspect

    - Aspect refers to the nature of the action, whether it is ongoing, completed, or repeated. It contributes to a more nuanced understanding of events in time.
    - Example: In English, "I am eating" (present progressive), "I have eaten" (present perfect), and "I eat" (simple present) convey different aspects of the action of eating.

By considering these linguistic features, analysts can disambiguate sentences, discern the intended meaning, and produce more accurate interpretations. Failure to account for these features may lead to misunderstandings, misinterpretations, or grammatical errors. In natural language processing and machine learning, the incorporation of features is fundamental for building robust models that can comprehend and generate human-like language.

Questions to Revise:

1. Short note on NLP with advantages, disadvantages and applications
2. Challenges in Natural Language Processing
3. Working Principle or Phases in Natural Language Processing
4. Grammars and parsing in Natural Language Processing
5. Features in linguistic analysis