18CSE359T Natural Language Processing

- Offered to Final Year B.Tech. CSE
- By Dept. of C.Tech.

Unit 1 Topics

- Introduction to Natural Language Processing
- Steps – Morphology – Syntax – Semantics
- Morphological Analysis (Morphological Parsing)
- Stemming – Lemmatization
- Parts of Speech Tagging

Unit 1 Topics …

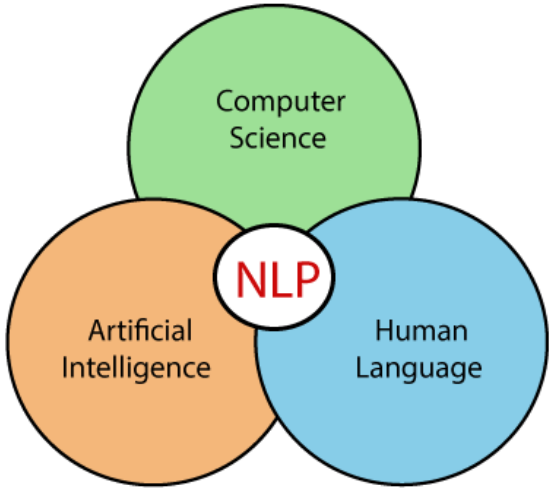
- Approaches on NLP Tasks (Rule-based, Statistical, Machine Learning)
- N-grams
- Multiword Expressions
- Collocations (Association Measures, Coefficients and Context Measures)
- Vector Representation of Words
- Language Modeling

Introduction to Natural Language Processing

Natural Language: Natural languages are languages used by human for communication. Example: Tamil, Telugu, Hindi, English, etc.

Natural Language Processing: NLP deals with processing, understanding and producing human languages by machine.

Need of processing human language: To develop different Human Computer Interactive (HCI) system through natural language.



Applications of NLP

- Information Retrieval system
- Text Summarization
- Speech Recognition
- Text to Speech
- Machine Translation System
- Question answering system

Issues and processing complexities of NLP

(i) Ambiguity

(ii) Language variability

(iii) Difficult to incorporate human cognition

Issues and processing complexities of NLP –
(i) Ambiguity

- **(i) Ambiguity:** Words in any natural languages usually have several different possible meanings.
- **Example bank→** financial Bank, Riverbank, Blood Bank
- For many NLP task, the proper sense of each ambiguous words in a sentence must be determine to interpret correct meaning.
- **Example:**
  - He saw (the boy) with a telescope.
  - He saw (the boy with a telescope).

Ambiguity

I made her duck.

- How many different interpretations does this sentence have?
- What are the reasons for the ambiguity?
- The categories of knowledge of language can be thought of as ambiguity resolving components.
- How can each ambiguous piece be resolved?
- Does speech input make the sentence even more ambiguous?
  - Yes – deciding word boundaries

Ambiguity (cont.)

- Some interpretations of : **`I made her duck.`**
  1. I cooked *duck* for her.
  2. I cooked *duck* belonging to her.
  3. I created a toy duck which she owns.
  4. I caused her to quickly lower her head or body.
  5. I used magic and turned her into a *duck*.
- `duck` – morphologically and syntactically ambiguous: noun or verb.
- `her` – syntactically ambiguous: dative or possessive.
- `make` – semantically ambiguous: cook or create.
- `make` – syntactically ambiguous:
  - Transitive – takes a direct object. => 2nd Interpretation
  - Di-transitive – takes two objects. => 5th
  - Takes a direct object and a verb. => 4th

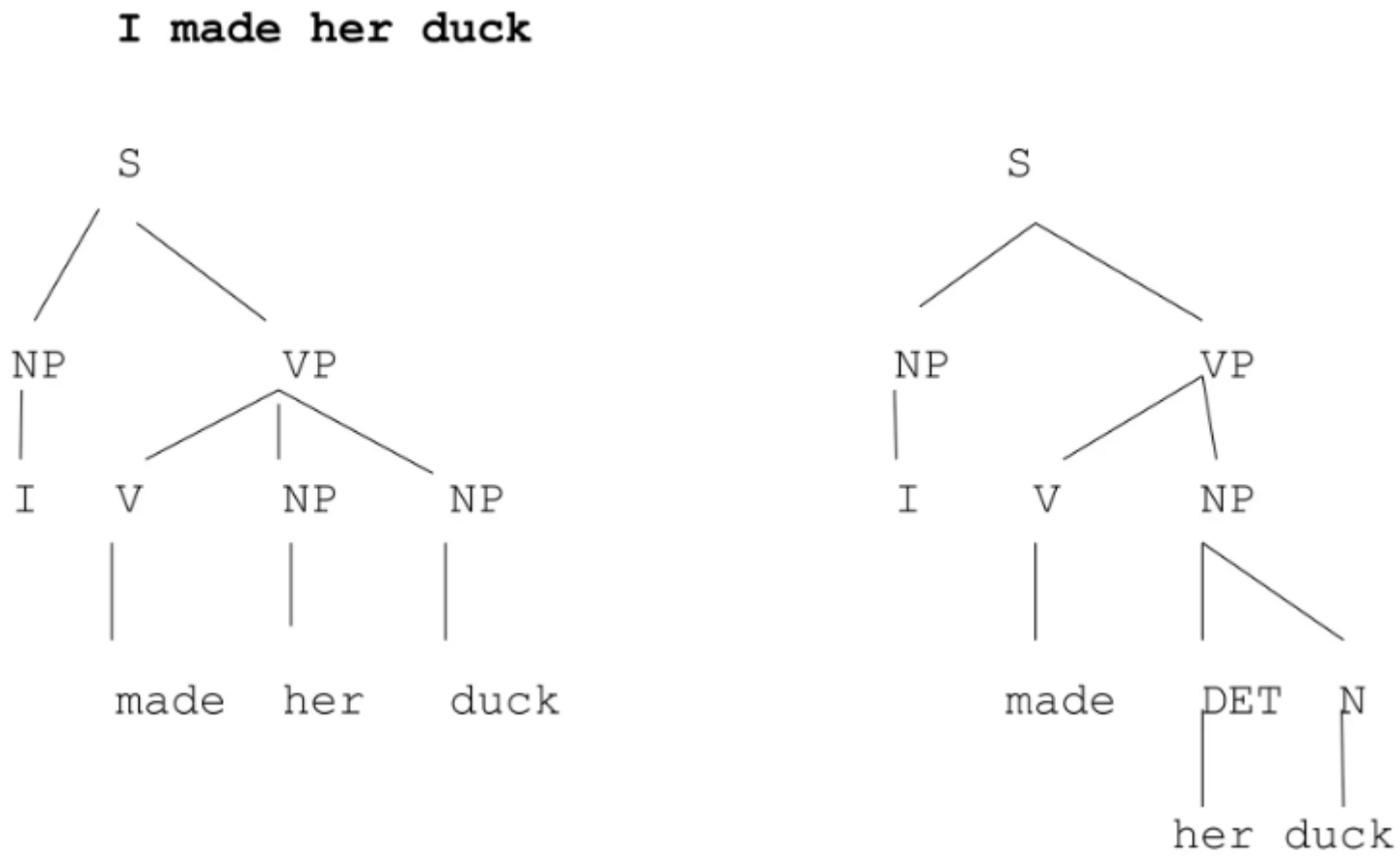
Ambiguity in a Turkish Sentence

- Some interpretations of: **Adamı gördüm.**
  1. I saw the man.
  2. I saw my island.
  3. I visited my island.
  4. I bribed the man.
- Morphological Ambiguity:
  - ada-m-ı ada+P1SG+ACC
  - adam-ı adam+ACC
- Semantic Ambiguity:
  - gör to see
  - gör to visit
  - gör to bribe

Resolve Ambiguities

- We will introduce *models* and *algorithms* to resolve ambiguities at different levels.
- **part-of-speech tagging** -- Deciding whether `duck` is verb or noun.
- **word-sense disambiguation** -- Deciding whether `make` is `create` or `cook`.
- **lexical disambiguation** -- Resolution of part-of-speech and word-sense ambiguities are two important kinds of lexical disambiguation.
- **syntactic ambiguity** -- `her duck` is an example of syntactic ambiguity, and can be addressed by probabilistic parsing.

Resolve Ambiguities (cont.)



Issues and processing complexities of NLP –
(ii) Language variability

- There are various ways to express the meaning of a word.
- Many languages are available worldwide and most of the languages have different character set, structure and grammar rules.
- It is difficult to design a language processing model that can capture a variety of language.
- **Example:** 22 official Indian Languages
- 3 different writings for Germany language (A pluri-centric language or polycentric language is a language with several codified standard forms, often corresponding to different countries)

Issues and processing complexities of NLP-
(iii) Difficult to incorporate human cognition

- Contextual Understanding
  - Human cognition involves understanding language within the broader context of the situation, background knowledge, and speaker intent.
  - Incorporating this contextual understanding into NLP systems is challenging because it requires models to grasp subtle nuances and infer unstated information.
- Pragmatics
  - Human communication often involves implicatures, where speakers convey meaning indirectly based on shared knowledge and cultural conventions.
  - Capturing pragmatics in NLP systems requires models to go beyond literal interpretations and infer implied meaning, which can be complex and context-dependent.

Issues and processing complexities of NLP-
(iii) Difficult to incorporate human cognition …

- Emotion and Tone
  - Human language is rich in emotional expression and tone, which significantly influence meaning and interpretation.
  - Incorporating emotional understanding into NLP systems involves detecting sentiment, tone, and other affective cues, which adds complexity to language processing tasks.
- Common Sense Reasoning
  - Humans rely on common sense knowledge and reasoning to understand language and make inferences.
  - Incorporating common sense reasoning into NLP systems is challenging because it requires models to possess a broad and deep understanding of the world, which is difficult to encode in computational systems.

Issues and processing complexities of NLP-
(iii) Difficult to incorporate human cognition …

- Dynamic Language Evolution
  - Human languages evolve over time due to cultural, social, and technological changes.
  - Incorporating the ability to adapt to evolving language usage and conventions into NLP systems is challenging because it requires models to continually learn and update their knowledge base.
- Ambiguity Resolution
  - Humans are adept at resolving ambiguities in language using context, background knowledge, and pragmatic cues.
  - Incorporating ambiguity resolution mechanisms into NLP systems is challenging because it requires models to accurately interpret ambiguous language in a wide range of contexts.

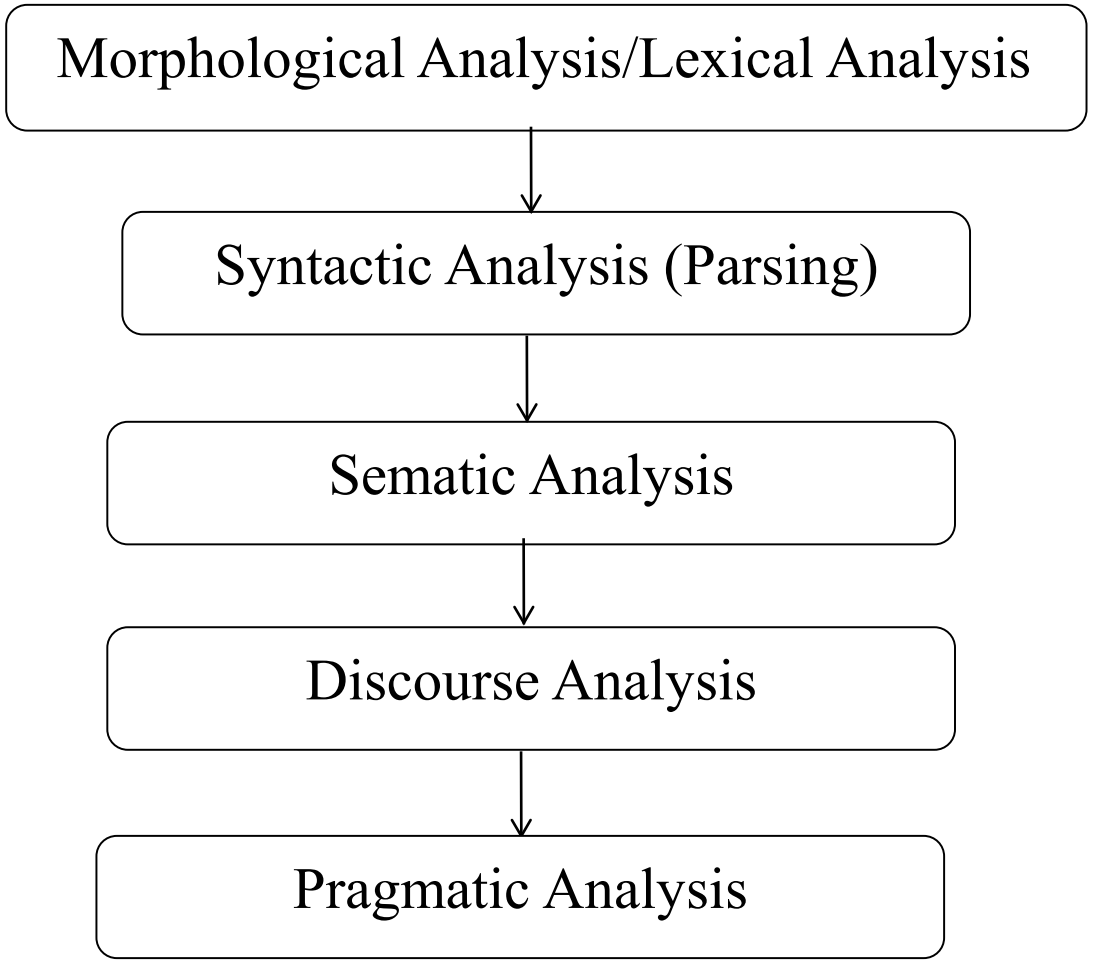
Overview of NLP

- NLP considers the hierarchical structure of a language where text represents set of symbols, several words make phrases, several phrases makes sentences that conveys the idea, and several sentences makes a paragraph that conveys some context.
- There are two main components of NLP
- **Natural language understanding**

  It involves mapping given input in natural language to useful representation.
- **Natural Language generation**

  It involves producing meaningful phrases and sentences to convey some internal representation

Phases of NLP



Phases of NLP

1.Morphological Analysis/ Lexical Analysis

2.Syntax Analysis

3.Semantic Analysis

4.Discourse

5.Pragmatics

Phases of NLP

(i) Morphological Analysis/Lexical Analysis

- It involves identifying and analyzing the structure of words.
- During lexical analysis, the whole text is divided into paragraphs, sentences and words.

(ii) Syntactic Analysis (Parsing)

- It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among them.
- Example: The school goes to boy {Rejected by syntactic analysis}

Phases of NLP …

(iii) Sematic Analysis

- It involves identifying the exact dictionary meaning from text.
- That is, it involves meaningful formation of sentences.
- Example: For example, the words 'write' and 'right'.
- They sound the same but mean different things.
- We can avoid confusion by choosing a different word, for example 'correct' instead of 'right'.

Phases of NLP …

(iv) Discourse Analysis

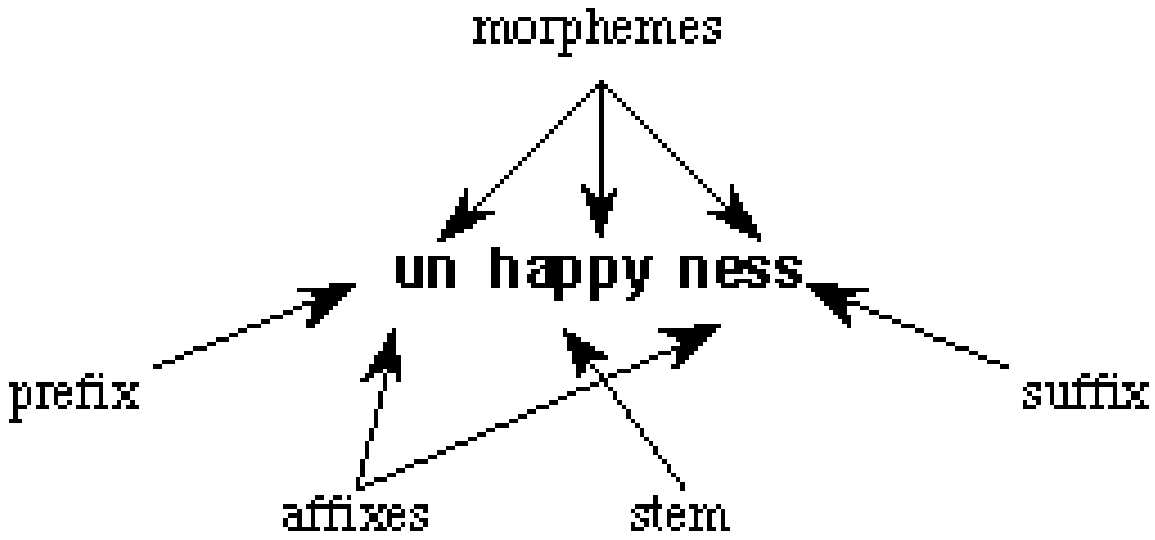
- The meaning of any sentence may depend on the meaning of the sentence that precedes it and may also influence the meaning of the sentence that follow it,
- Example: She wanted it. (Not clear unless we are aware of the previous sentence)

(v) Pragmatic Analysis

- The sentences representing what it said is reinterpreted to determine what was meant.
- It involve deriving aspects of language which require real world knowledge.
- Example: Do you know who am I? (Machine may not understand the expression behind the sentence)

Morphological Analysis

- Morphology is the study of the structure and formation of words.
- Its most important unit is the morpheme, which is defined as the "minimal unit of meaning".
- Consider a word like: "unhappiness". This has three parts:



Goal of morphological parser

- The goal of morphological parser is to discover the morpheme that is behind a given word.
  - cats→ cat+s
  - cat is the root or stem word
  - s is the plural
- Morphological analysis and parsing are essential for many NLP applications such as
  - spelling error detection,
  - translation and
  - information retrieval etc.,

Broad classes of morphemes

There are two broad classes of morphemes

- **Stem:** The main morpheme that contains the real meaning
- **Affixes:** Modifies the meaning given by the stem

Affixes - Types

Affixes can be of 4 types

- **Prefix:** morphemes appear before a stem
- **Suffix:** morphemes applied to the end of the stem
- **Infix:** morpheme that appear in between the stem
- **Circumfix:** morphemes applied before and after a stem

Different ways to form words

- There are three different ways to form words

1. Inflection
2. Derivation
3. Compounding

1. Inflection

- Inflection is the process of changing the form of a word so that it expresses information such as number, person, case, gender, tense, mood and aspect, but the syntactic category of the word remains unchanged.
- As an example, the plural form of the noun in English is usually formed from the singular form by adding an s.
  - car / cars
  - table / tables
  - dog / dogs
- In each of these cases, the syntactic category of the word remains unchanged.

2. Derivation

- Inflection doesn't change the syntactic category of a word.
- Derivation does change the category.
- Combines word stem with a grammatical morpheme resulting a word belonging to a different class.
- Example:
  - Teacher→ teaching
  - (noun)→(verb)

3. Compounding

- Process of merging two or more words to form a new word.
- Example: overlook, desktop, download

Information sources used by morphological parser

- A morphological parser uses the following information sources
- **Lexicon:** A lexicon list contains stems and affixes together with basic information about them.
- **Morphotactics:**
- It includes the allowable ordering of morphemes that constitutes a word.
- "the set of rules that define how morphemes (morpho) can touch (tactics) each other".
- Many English affixes may only be attached directly to morphemes with particular parts of speech:
- do + -able + -ity = doability
- but not
- do + -ity + -able = *doityable
- The suffix -ity produces a noun from an adjective, and -able creates adjectives from verbs.[1] To reverse the order violates the rules of English morphotactics, making the word ungrammatical (marked with an asterisk).
- **Orthographic rules:** These rules specify the spelling rules that specifies the changes that occur when two given morphemes are combined.

  Example: y→ ier

  Easy→ easier

Stemming and Lemmatization

- Stemming and lemmatization are both techniques used in natural language processing (NLP) to reduce words to their base or root forms.
- While they serve a similar purpose, they operate differently and have distinct advantages and disadvantages.
- While both stemming and lemmatization serve the purpose of reducing words to their base forms, lemmatization tends to be more accurate as it considers the context and meaning of words, while stemming is faster and simpler but may lead to loss of meaning in certain cases.

Stemming and Lemmatization – When to choose?

- Stemming
  - Choose stemming when speed and simplicity are prioritized over accuracy, or
  - when working with tasks like information retrieval where minor variations in word forms are acceptable.
- Lemmatization
  - Opt in for lemmatization when accuracy and semantic meaning preservation are critical, such as in text analysis tasks where precise understanding of the text is required.

Stemming

- Stemming is the process of reducing words to their word stems or root forms by removing suffixes or prefixes.
- Algorithmic Approach: Stemming algorithms chop off the ends of words based on common prefixes or suffixes, without necessarily understanding the context or meaning of the word.
- Example: For instance, "running" might be stemmed to "run".

Stemming …

- Advantages
  - Simplicity and speed: Stemming algorithms are typically faster and less resource-intensive than lemmatization.
  - Useful for tasks where speed is crucial and minor inaccuracies are acceptable, such as search engines.
- Disadvantages
  - Stemmed words may not always be valid words in the language, leading to potential loss of meaning or grammatical errors.
  - Stemmed forms may not necessarily be semantically related to the original word.

Simplest morphological systems: Stemmers

- Morphological analysis can be avoided if an existing lexicon is available that list features for all word forms of all root words in a language.
- However, it is not practical to list all possible combination of word forms in many languages.
- The simplest morphological systems are stemmers that converts morphological variations of a given word to its stem.
- Stemmers does not require lexicon instead it uses a set of rewrite rules to derive the root word.

  ien→y

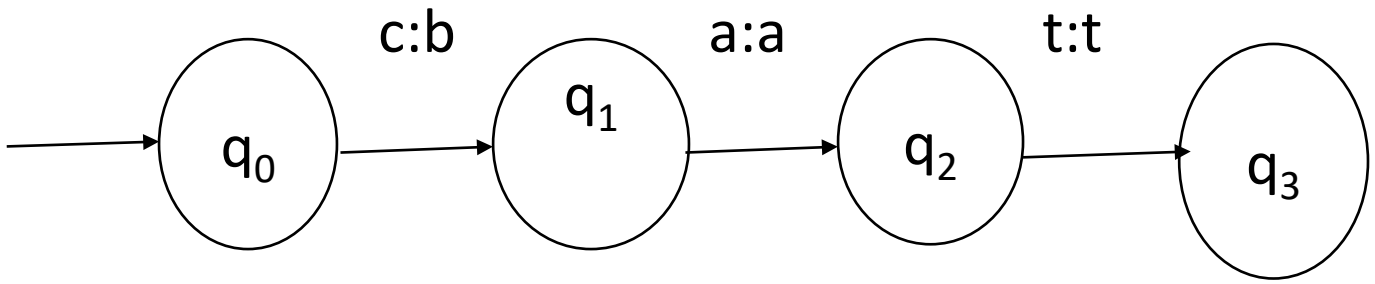
  ing→e

  Stemmer: Example: rotational→rotate

  rule: ational→ate

Stemming algorithm: Working

- Stemming algorithm works in two steps
- Suffix removal: Removes pre-defined endings from words
- Recording: Add pre-defined endings to the output of the 1st step.
- This model is implemented with special type of finite state automata called finite state transducer. Transducer maps a set of symbols to another. To replace cat by bat



Lemmatization

- Lemmatization, on the other hand, is the process of reducing words to their base or canonical forms, known as lemmas, by considering the word's context and meaning.
- Algorithmic Approach: Lemmatization algorithms utilize language dictionaries and morphological analysis to properly reduce words to their dictionary forms.
- Example: For example, "better" would be lemmatized to "good", as it's the base form of the word.

Lemmatization …

- Advantages
  - Retains the semantic meaning of the word, producing valid words in the language.
  - Useful for tasks where accuracy and understanding of the text are crucial, such as sentiment analysis or machine translation.
- Disadvantages
  - Lemmatization is generally slower and more complex compared to stemming.
  - Requires more computational resources and linguistic knowledge, as it involves dictionary lookups and morphological analysis.

Syntax Analysis

- Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar.
- The objective of syntactic analysis is to find the syntactic structures of the sentence that determines the meaning of the sentence.
- It can be represented as a tree where nodes represent phrases, leaf nodes are words.

NP- Noun Phrase

- NP is a phrase whose head is noun or pronoun optionally accomplished by a set of modifiers.
- NP→ Noun
- NP→ Pronoun
- NP→ Det Noun
- NP→ Adj Noun
- NP→ Det Adj Noun

VP- Verb Phrase

- The verb phrase can be a single verb, a verb followed by a noun phrase, or a verb followed by a prepositional phrase.
- The production rules for a VP:
  - VP→ Verb
  - VP→ Verb NP
  - VP→ Verb PP

Sentence and Question Structure

Sentence

- S→ VP
- S→ NP VP

Wh question Structure

- S→Wh NP VP
- S→Wh NP Aux NP VP

Yes or No Question

- S→Aux NP VP

Top-Down Parsing

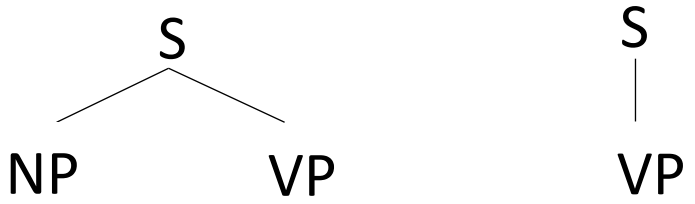
- In this kind of parsing, the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input.
- The most common form of top-down parsing uses recursive procedure to process the input.
- Top-down parsing starts its search from the root node and works downwards towards the leaf node.
- The root is expanded using the grammar with 'S' as non-terminal symbols.
- Each non-terminal symbol in the resulting sub-tree is then expanded using the appropriate grammar rules.

S→NP VP	**NP→Noun**	**Preposition→from|with|on|to**
S→VP	**VP→Verb**	**Verb→plays|point**
NP→ Det Noun	**VP→Verb Noun**	**Noun→Student|..**
NP→ pronoun	**PP→ Prep NP**	
NP→ Det Noun PP	**Det→This|That|the**	
	Pronoun→ She|He|they	

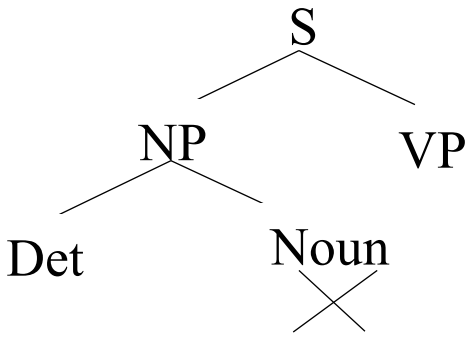
Example 1: Paint the door

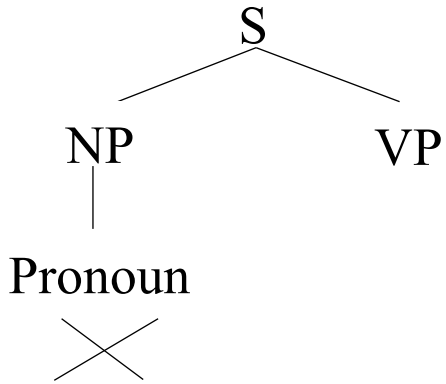
Level 1: S

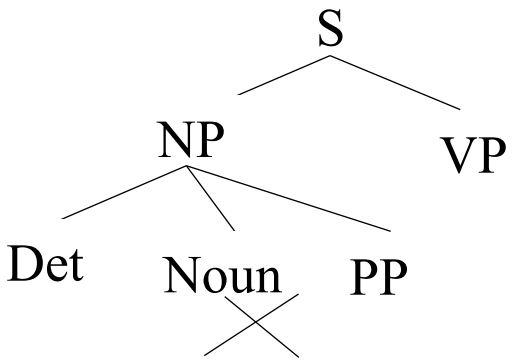
Level 2:

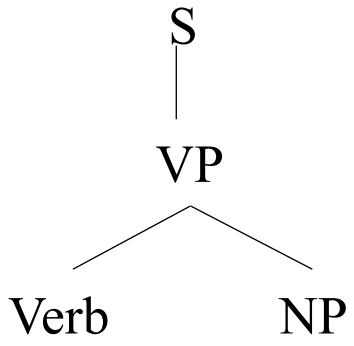


Level 3:

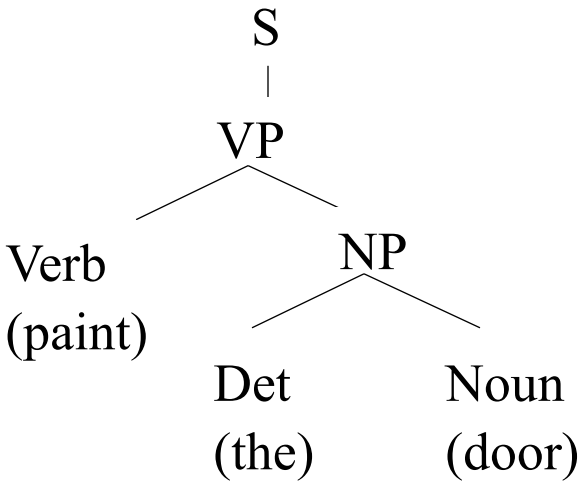




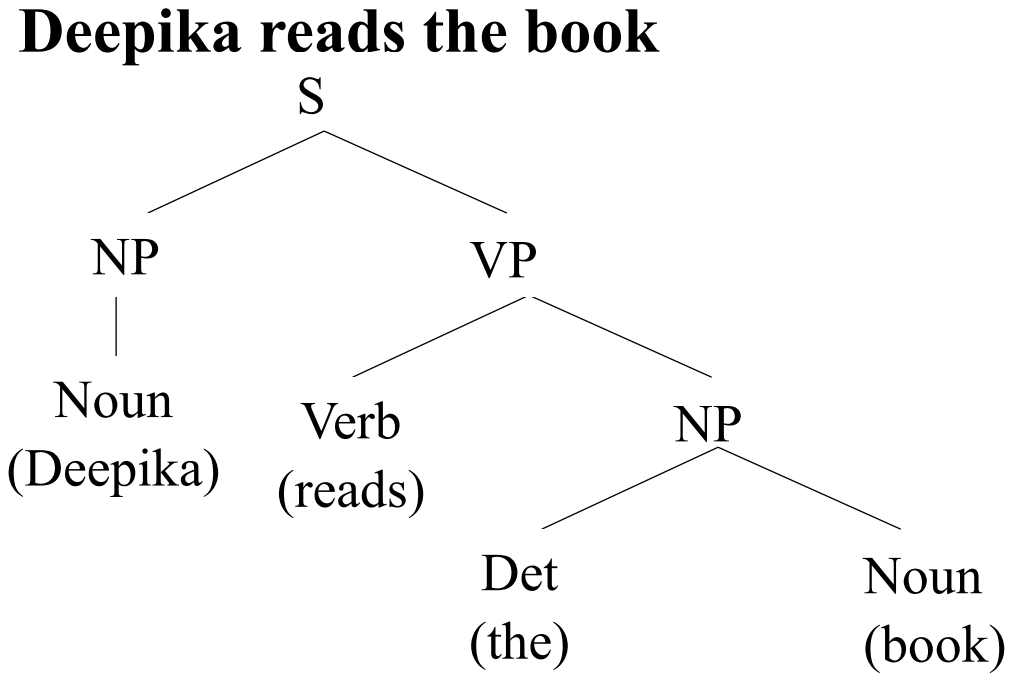




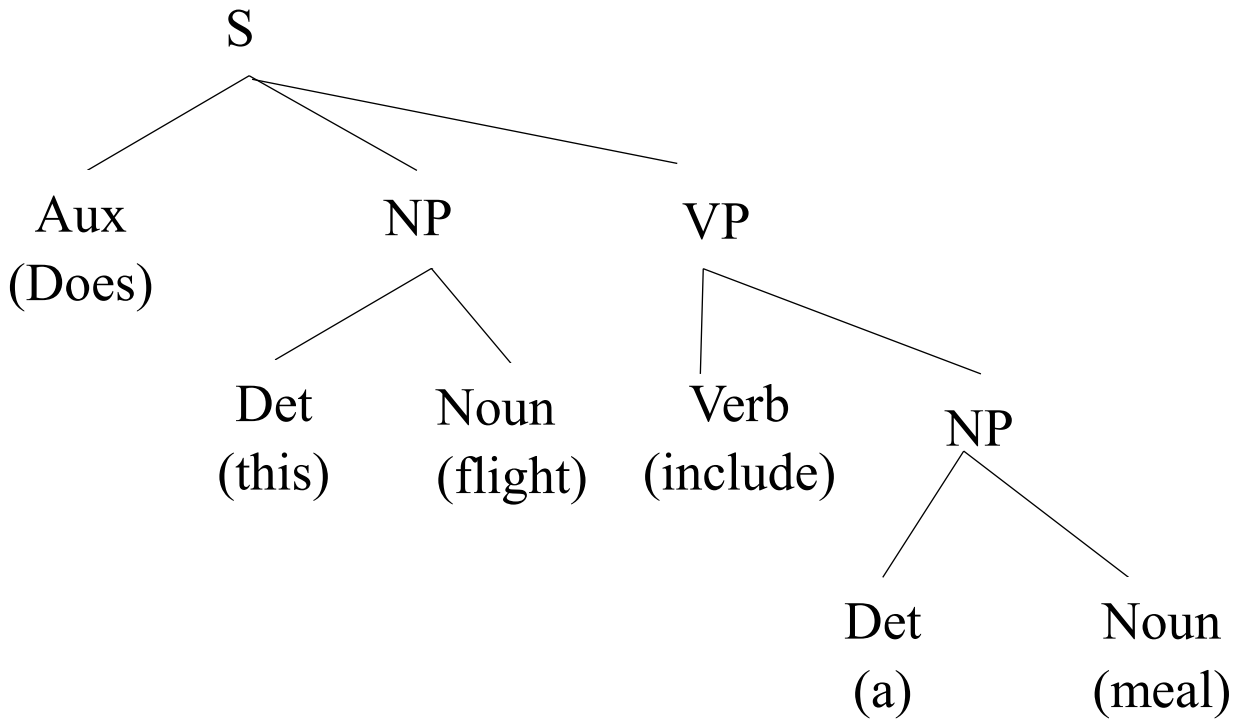
Level 4:



Example 2: Deepika reads the book



Example 3: Does this flight include a meal?



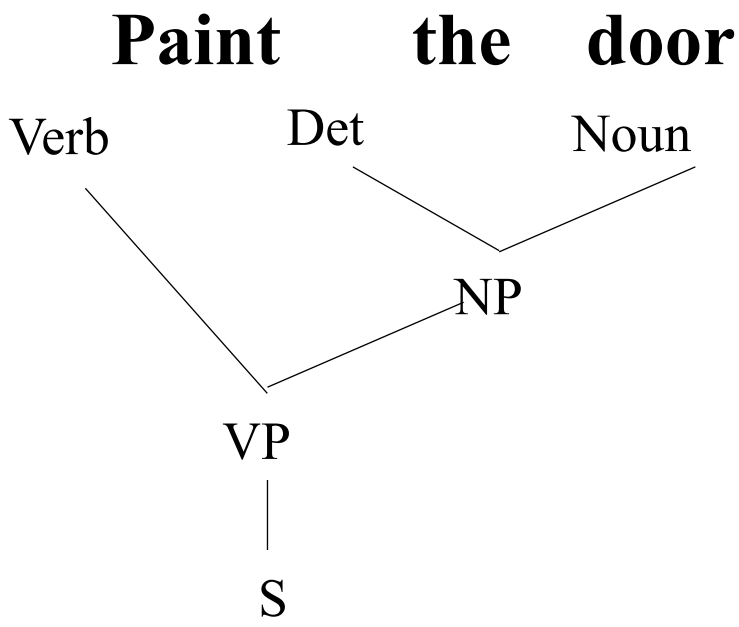
Advantages and Disadvantages of Top-Down parsing

- **Advantages**
- There is a chance of generating wrong grammatical sentence as it starts generating the tree using the start symbol of grammar.
- **Disadvantages**
- Time consuming because it checks each and every rule of parsing

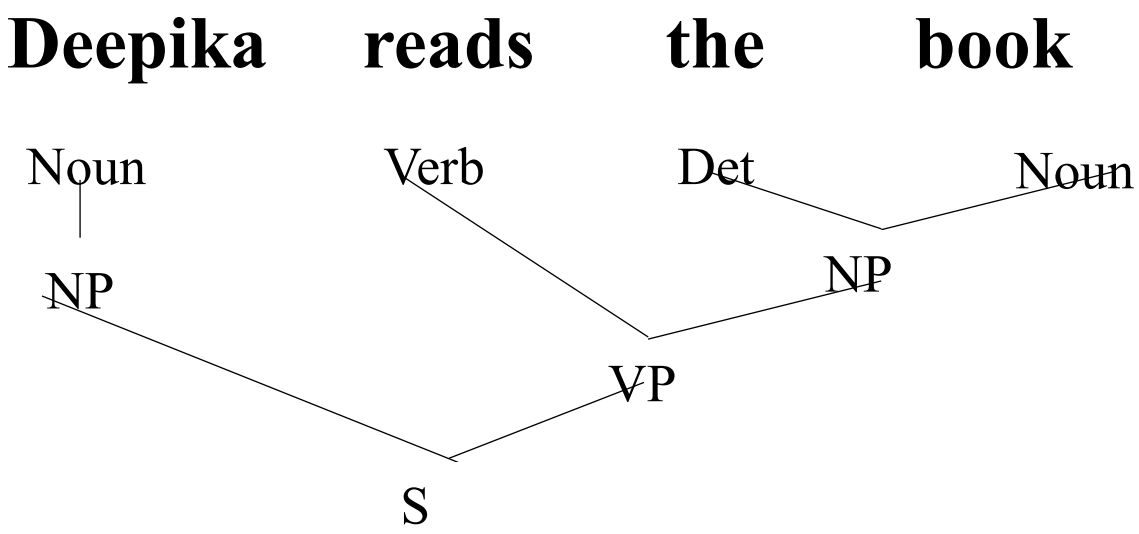
Bottom-Up Parsing

- In this kind of parsing, the parser starts with the input symbol and tries to construct the parse tree in an upward direction towards the root.
- At each step the parser checks the rules in the grammar where the RHS matches the portion of the parse tree constructed so far.
- It then reduces it using the LHS of the production.
- The parse tree is considered to be successful if the parser reduces the tree to the start symbol of the grammar.

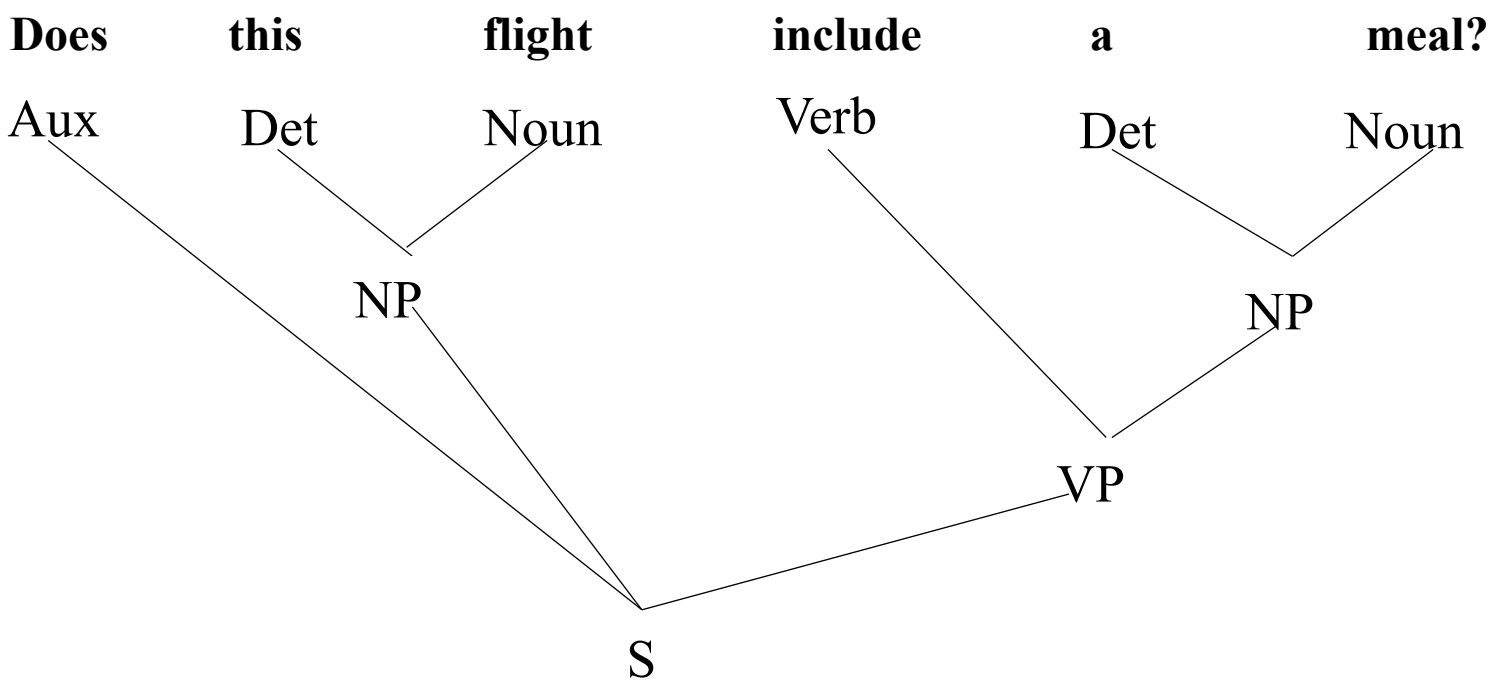
Example 1: Paint the door



Example 2: Deepika reads the book



Example 3: Does this flight include a meal?



Advantages and Disadvantages of Bottom-Up Parsing

- **Advantages**

- It never wastes time in exploring a tree that does not match the input.
- **Disadvantages**
- It wastes time in generating trees that have no chance of leading to S rooted tree.

Disadvantages of parsing

- Left Recursion Leading to Infinite Loops:
  - Top-down parsers cannot handle left-recursive grammars directly, as they result in non-terminating recursive calls.
- Ambiguity:
  - Ambiguous grammars can lead to multiple valid parse trees for a single input, complicating the parsing process and making it difficult to determine the intended structure and meaning.
  - Addressing these disadvantages involves transforming grammars to remove left recursion and refine them to resolve ambiguities, ensuring that parsers can operate efficiently and accurately.

Semantic Analysis

- It involves mapping of natural language occurrences to some representation of meaning.
- A meaning representation language bridges the gap between linguistic and common-sense knowledge.
- Example:
- A cup on the table→ on(cup, table)
- Air India serves Hyderabad→ serves(Air India, Hyderabad)

Example of meaning representation language are

- First Order Predicate Calculus
- Semantic Network
- Conceptual Dependencies

Characteristics of Meaning representation language

- **Verifiable:** We should be able to identify the meaning and also verify the meaning.
- **Unambiguous:** There should not be any ambiguity. There should be only one representation of the sentence.
- **Support canonical form**
  - Does Air India serves Hyderabad?- →serves(x,y)
  - Does Air India offers a flight to Hyderabad?-→ offers(x,y)
  - Does Air India have a flight to Hyderabad?- →have (x,y)
- **Support inference and variables**
- **Expressiveness:** A wide range of domain must be served.

Meaning representation creation and assignment to linguistic units

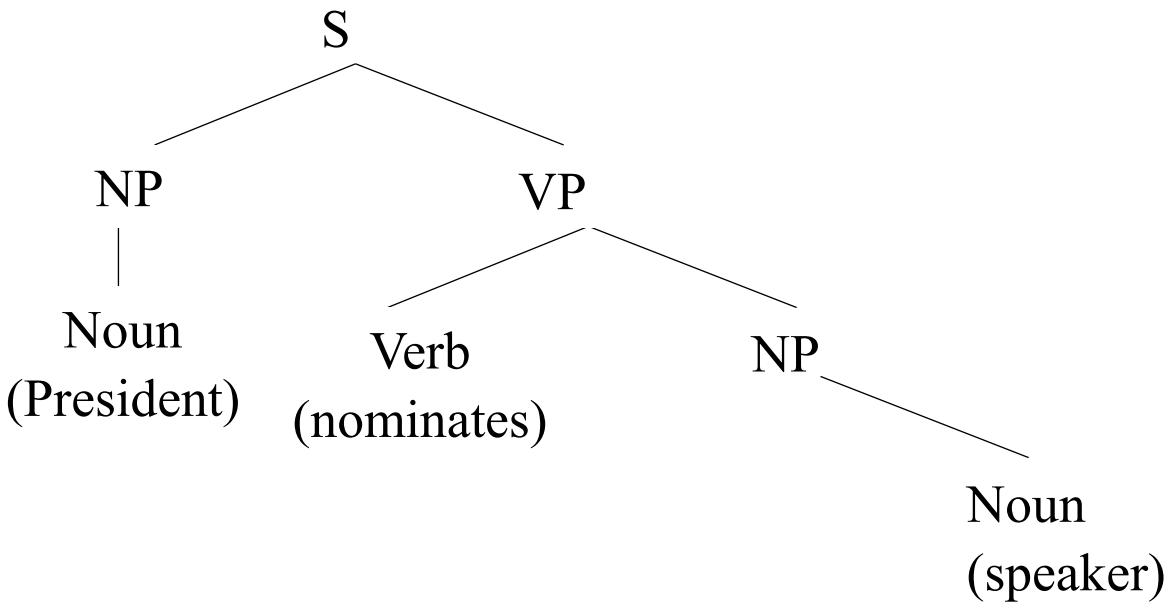
- Meaning representation can be created and assigned to linguistic units using two approaches

(i) Syntax driven semantic analysis

(ii) Semantic Analysis

(i) Syntax driven semantic analysis

- It uses syntactic constituents of a sentence to build its meaning representation.
- Example: President nominates speaker



(i) Syntax driven semantic analysis …

- It is based on the "principle of compositionality" that states that meaning of whole sentence can be derived from the meaning of its parts.
- In reality the actual meaning of sentence can be derived from constituent meaning, relationship between words, context, domain, word order and real-world knowledge.
  - The old man finally kicked the bucket-→ The man finally died (idiom)
  - All roads lead to Rome-→ There are many ways to do a task. (idiom)

- **Disadvantages**
- It depends upon traditional grammar rules which aren't meant for semantic processing.
- It only provides the abstract information which many times does not fit the semantic meaning.

(ii) Semantic Grammar

- Example: I want to go from Delhi to Chennai on 24th March.
- Info request: User wants to go to city from city Time Expr

Drawback

- It is domain dependent. Whenever we are adding new domain we have to write new rule.
- It focusses on building semantic capabilities in to grammar
- Here, rules and constituents corresponds directly to entities and activities in the domain
- Rules are domain specific, therefore it is required to develop new rules for each domain
- Example: Indian Hotel and Chinese Food→ We need to write different rules for this phase because they belong to two different context.

Applications of semantic analysis

- Information retrieval
- Machine translation
- Question answering system
- Sentimental analysis
- Text classification
- Text analysis

Importance of semantic analysis

- It is concerned with understanding the actual meaning by extracting contextual information.
- Example: Feeling hungry→ nearby restaurants
- **Word net:** It is a large lexical data base for English language that list all senses of a word.
- Relationships are 4 types namely:
  - **Homonymy**: words with different meaning
  - **Polysemy**: Many possible meaning for a word
  - **Antonymy:** opposite of a word
  - **Synonymy**: Many word for single meaning

Part of Speech (POS) tagging

- Part of Speech (POS) tagging is the process of assigning a part of speech to each word in a sentence.
- Parts of speech include categories such as nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, and interjections.
- POS tagging is a fundamental task in natural language processing (NLP) as it provides essential information for various downstream applications such as syntactic parsing, named entity recognition, and machine translation.
- These POS tags provide information about the grammatical structure of each sentence, which can be useful for various natural language processing tasks such as syntactic analysis, information extraction, and machine translation.
- POS tagging or PoS tagging or POST, is also called grammatical tagging

Example Sentence and POS Tags

- Consider the sentence: "The quick brown fox jumps over the lazy dog."
- The POS tags for this sentence would be:
  - The (Determiner, DT)
  - quick (Adjective, JJ)
  - brown (Adjective, JJ)
  - fox (Noun, NN)
  - jumps (Verb, VBZ)
  - over (Preposition, IN)
  - the (Determiner, DT)
  - lazy (Adjective, JJ)
  - dog (Noun, NN)

Example 2: Applying POS Tag to "Our dog chased a brown cat away from the home"

- To apply Part of Speech (POS) tagging to the given text, we will tag each word with its corresponding POS tag.
- Here's the tagging for the sentences provided:
  - **Our** (PRP$ - Possessive pronoun)
  - **dog** (NN - Noun, singular or mass)
  - **chased** (VBD - Verb, past tense)
  - **a** (DT - Determiner)
  - **brown** (JJ - Adjective)
  - **cat** (NN - Noun, singular or mass)
  - **away** (RB - Adverb)
  - **from** (IN - Preposition or subordinating conjunction)
  - **the** (DT - Determiner)
  - **home** (NN - Noun, singular or mass)

Example 3: Applying POS Tag to "The boy put the dogs in the bag"

The boy put the dogs in the bag

- The - DT (Determiner)
- boy - NN (Noun, singular or mass)
- put - VBD (Verb, past tense)
- the - DT (Determiner)
- dogs - NNS (Noun, plural)
- in - IN (Preposition or subordinating conjunction)
- the - DT (Determiner)
- bag - NN (Noun, singular or mass)

Closed and open classes

- Part of Speech (POS) tags can be categorized into
  - closed and open classes
  - based on whether new words can be easily added to the class.

Closed POS Tag Classes

- Closed classes contain a limited set of words, and new words are rarely added.
- These classes generally serve grammatical purposes rather than carrying content.
- Pronouns
- PRP: Personal pronoun (e.g., "I", "she", "they")
- PRP$: Possessive pronoun (e.g., "my", "her", "their")
- WP: Wh-pronoun (e.g., "who", "what")
- WP$: Possessive wh-pronoun (e.g., "whose")
- Determiners
- DT: Determiner (e.g., "a", "the", "an")
- PDT: Predeterminer (e.g., "all", "both", "half")
- WDT: Wh-determiner (e.g., "which", "that", "whatever")
- Conjunctions
- CC: Coordinating conjunction (e.g., "and", "but", "or")
- IN: Preposition or subordinating conjunction (e.g., "in", "from", "that")

Closed POS Tag Classes …

- Prepositions and Particles
- IN: Preposition or subordinating conjunction (e.g., "in", "on", "under")
- RP: Particle (e.g., "up" in "give up", "off" in "take off")
- Auxiliary Verbs and Modals
- MD: Modal (e.g., "can", "will", "should")
- TO: to (infinitive marker, e.g., "to go", "to play")
- Existential There
- EX: Existential there (e.g., "There is a problem")
- Interjections
- UH: Interjection (e.g., "oh", "wow")

Open POS Tag Classes

- Open classes can easily accept new words.
- These classes are where most of the vocabulary expansion in a language occurs, as new terms are coined and borrowed.
- Nouns
- NN: Noun, singular or mass (e.g., "girl", "tennis")
- NNS: Noun, plural (e.g., "girls", "dogs")
- NNP: Proper noun, singular (e.g., "Kavya", "London")
- NNPS: Proper noun, plural (e.g., "Americans", "Smiths")
- Verbs
- VB: Verb, base form (e.g., "permit", "play")
- VBD: Verb, past tense (e.g., "played", "refused")
- VBG: Verb, gerund or present participle (e.g., "playing", "refusing")
- VBN: Verb, past participle (e.g., "played", "refused")
- VBP: Verb, non-3rd person singular present (e.g., "play", "am", "refuse")
- VBZ: Verb, 3rd person singular present (e.g., "plays", "is", "permits")

Open POS Tag Classes …

- Adjectives
- JJ: Adjective (e.g., "intelligent", "quick")
- JJR: Adjective, comparative (e.g., "quicker", "better")
- JJS: Adjective, superlative (e.g., "quickest", "best")
- Adverbs
- RB: Adverb (e.g., "quickly", "well")
- RBR: Adverb, comparative (e.g., "more quickly", "better")
- RBS: Adverb, superlative (e.g., "most quickly", "best")
- Foreign Words and Symbols
- FW: Foreign word (e.g., "de facto")
- SYM: Symbol (e.g., "$", "%")
- List Item Markers
- LS: List item marker (e.g., "1.", "A.")

Closed and open classes: Examples in Context

- Closed Class Example Sentence:
- "The quick brown fox jumps over the lazy dog."
- "The" (DT - Determiner)
- "over" (IN - Preposition)
- "the" (DT - Determiner)
- Open Class Example Sentence:
- "The quick brown fox jumps over the lazy dog."
- "quick" (JJ - Adjective)
- "brown" (JJ - Adjective)
- "fox" (NN - Noun, singular)
- "jumps" (VBZ - Verb, 3rd person singular present)
- "lazy" (JJ - Adjective)
- "dog" (NN - Noun, singular)
- These categorizations help in understanding the flexibility and roles of different types of words in a language.

POS Tag set Examples

- Brown corpus tagset (87 tags)
- Penn Treebank tagset (45 tags)
- C7 tagset (146 tags)

Penn Treebank Tagset

- An example of a POS tagset using the Penn Treebank Tagset which is one of the most widely used tagsets in natural language processing
- It has 45 tags
- Example tags
  - CC: Coordinating conjunction
  - CD: Cardinal number
  - DT: Determiner
  - EX: Existential there
  - FW: Foreign word
  - IN: Preposition or subordinating conjunction
  - JJ: Adjective
  - JJR: Adjective, comparative
  - JJS: Adjective, superlative
  - LS: List item marker
  - MD: Modal

Penn Treebank Tagset …

- NN: Noun, singular or mass
- NNS: Noun, plural
- NNP: Proper noun, singular
- NNPS: Proper noun, plural
- PDT: Predeterminer
- POS: Possessive ending
- PRP: Personal pronoun
- PRP$: Possessive pronoun
- RB: Adverb
- RBR: Adverb, comparative
- RBS: Adverb, superlative
- RP: Particle
- SYM: Symbol

Penn Treebank Tagset …

- TO: to
- UH: Interjection
- VB: Verb, base form
- VBD: Verb, past tense
- VBG: Verb, gerund or present participle
- VBN: Verb, past participle
- VBP: Verb, non-3rd person singular present
- VBZ: Verb, 3rd person singular present
- WDT: Wh-determiner
- WP: Wh-pronoun
- WP$: Possessive wh-pronoun
- WRB: Wh-adverb
- This tagset provides a comprehensive set of POS tags to annotate the words in a sentence, allowing for detailed grammatical analysis and processing.

Brown Corpus Tagset

- The Brown Corpus is a well-known corpus of English-language text, and it comes with its own tagset for annotating parts of speech.
- It has 87 tags
- Here are examples of tagsets used in the Brown Corpus

  1.AT: Article (a, the)

  2.NP: Noun, singular or mass (e.g., "cat", "house")

  3.NNS: Noun, plural (e.g., "cats", "houses")

  4.VB: Verb, base form (e.g., "run", "eat")

  5.VBD: Verb, past tense (e.g., "ran", "ate")

  6.VBG: Verb, gerund or present participle (e.g., "running", "eating")

Brown Corpus Tagset …

7. **VBN**: Verb, past participle (e.g., "run", "eaten")
8. **JJ**: Adjective (e.g., "quick", "brown")
9. **RB**: Adverb (e.g., "quickly", "well")
10. **CC**: Coordinating conjunction (e.g., "and", "but")
11. **IN**: Preposition or subordinating conjunction (e.g., "in", "on")
12. **TO**: to (e.g., "to go", "to eat")

- These are just a few examples of the tags used in the Brown Corpus.
- The tagset includes additional tags for other parts of speech, punctuation, and special symbols.

Applying PoS tagging to sentences

- Apply PoS tagging to the following sentences in Table Format: Word, PoS Type, Explanation
  - I am a girl
  - Kavya is an intelligent girl
  - She plays tennis
  - They play football
  - They refuse to permit us

Applying PoS tagging to sentences: Example 1 - I am a girl

Word	POS Type	Explanation
I	PRP	Personal pronoun
am	VBP	Verb, non-3rd person singular present
a	DT	Determiner
girl	NN	Noun, singular or mass

Automated POS tagging

- Automated POS tagging is the process of using computational algorithms to assign parts of speech to each word in a given text.
- This task is essential in natural language processing (NLP) for understanding the grammatical structure of sentences.

-

How Automated POS Tagging Works?

- Here's a brief overview of how automated POS tagging works and some of the popular tools and techniques used:

1. Tokenization: The text is first tokenized into individual words or tokens.

- For example, the sentence "The quick brown fox jumps over the lazy dog" is split into tokens: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"].

How Automated POS Tagging Works? …

2. Tagging Algorithms

- Several algorithms can be used to tag each token with its corresponding POS tag.
- Some of the most common algorithms include:
  - Rule-Based Tagging: Uses a set of predefined linguistic rules.
  - Statistical Tagging: Uses probabilistic models trained on annotated corpora (e.g., Hidden Markov Models).
  - Machine Learning-Based Tagging: Uses classifiers such as Decision Trees, Maximum Entropy Models, and Neural Networks.

3. Context Consideration

- Modern POS taggers consider the context of each word to improve accuracy.
- This is especially important for words that can serve multiple parts of speech depending on their usage (e.g., "can" as a noun or a verb).

Steps Involved in the POS tagging

•**Collect a dataset of annotated text:** This dataset will be used to train and test the POS tagger.
•The text should be annotated with the correct POS tags for each word.
•**Preprocess the text:** This may include tasks such as tokenization (splitting the text into individual words),
•lowercasing, and removing punctuation.
•**Divide the dataset into training and testing sets:** The training set will be used to train the POS tagger,
• and the testing set will be used to evaluate its performance.
•**Train the POS tagger:** This may involve building a statistical model, such as a hidden Markov model (HMM),
•or defining a set of rules for a rule-based or transformation-based tagger.
•The model or rules will be trained on the annotated text in the training set.
•**Test the POS tagger:** Use the trained model or rules to predict the POS tags of the words in the testing set.
•Compare the predicted tags to the true tags and calculate metrics such as precision and recall to evaluate
•the performance of the tagger.
•**Fine-tune the POS tagger:** If the performance of the tagger is not satisfactory, adjust the model or rules and
•repeat the training and testing process until the desired level of accuracy is achieved.
•**Use the POS tagger:** Once the tagger is trained and tested, it can be used to perform POS tagging on new,
•unseen text.

Different POS Tagging Techniques

- Rule-Based Tagging
- Statistical Tagging
- Transformation-Based Tagging
- Machine Learning-Based Tagging

1.Rule-Based POS Tag

- This is one of the oldest approaches to POS tagging.
- It involves using a dictionary consisting of all the possible POS tags for a given word.
- If any of the words have more than one tag, **hand-written rules** are used to assign the correct tag based on the tags of surrounding words.
- For example, if the preceding of a word an article, then the word has to be a noun.
- Consider the words: *A Book*
- **Get all the possible POS tags for individual words:** A – Article; Book – Noun or Verb
- **Use the rules to assign the correct POS tag:** As per the possible tags, "A" is an *Article* and we can assign it directly. But, a book can either be a Noun or a Verb. However, if we consider "A Book", A is an article and following our rule above, Book has to be a *Noun*. Thus, we assign the tag of Noun to book.
- **POS Tag:** [("A", "Article"), ("Book", "Noun")]

1.Rule-Based POS Tag …

- These rules may be either −
- **Context-pattern rules**
- **Regular expression** compiled into finite-state automata, intersected with lexically ambiguous sentence representation.
- Rule-based POS tagging by its two-stage architecture −
- **First stage** − In the first stage, it uses a dictionary to assign each word a list of potential parts-of-speech.
- **Second stage** − In the second stage, it uses large lists of hand-written disambiguation rules to sort down the list to a single part-of-speech for each word.

1.Rule-Based POS Tag …

- These taggers are knowledge-driven taggers.
- The rules in Rule-based POS tagging are built manually.
- The information is coded in the form of rules.
- We have some limited number of rules approximately around 1000.
- Smoothing and language modeling is defined explicitly in rule-based taggers.

Rule-based POS tagger Example

1. Define a set of rules for assigning POS tags to words. For example:
- If the word ends in “-tion,” assign the tag “noun.”
- If the word ends in “-ment,” assign the tag “noun.”
- If the word is all uppercase, assign the tag “proper noun.”
- If the word is a verb ending in “-ing,” assign the tag “verb.”

2. Iterate through the words in the text and apply the rules to each word in turn. For example:
- “Nation” would be tagged as “noun” based on the first rule.
- “Investment” would be tagged as “noun” based on the second rule.
- “UNITED” would be tagged as “proper noun” based on the third rule.
- “Running” would be tagged as “verb” based on the fourth rule.

3. Output the POS tags for each word in the text.

Statistical POS Tagging

- Statistical part-of-speech (POS) tagging is a method of labeling words with their corresponding parts of speech using statistical techniques.
- This contrasts with rule-based POS tagging, which relies on pre-defined rules, and unsupervised learning-based POS tagging, which does not use any annotated training data.
- In statistical POS tagging, a model is trained on a large annotated corpus of text to learn the patterns and characteristics of different parts of speech.
- The model uses this training data to predict the POS tag of a given word based on the context in which it appears and the probability of different POS tags occurring in that context.
- Statistical POS taggers can be more accurate and efficient than rule-based taggers, especially for tasks with large or complex datasets.

Stochastic POS Tagging

- Stochastic POS tagging is a subset of statistical POS tagging that specifically uses frequency or probability models.
- Any approach to part-of-speech tagging that includes statistical methods can be referred to as stochastic tagging.
- Another technique of tagging is Stochastic POS Tagging. Now, the question that arises here is which model can be stochastic.
- The model that includes frequency or probability (statistics) can be called stochastic.
- Any number of different approaches to the problem of part-of-speech tagging can be referred to as stochastic tagger.
- The simplest stochastic tagger applies the following approaches for POS tagging −
- **Word Frequency Approach**
- In this approach, the stochastic taggers disambiguate the words based on the probability that a word occurs with a particular tag. We can also say that the tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word. The main issue with this approach is that it may yield inadmissible sequence of tags.
- **Tag Sequence Probabilities**
- It is another approach of stochastic tagging, where the tagger calculates the probability of a given sequence of tags occurring. It is also called n-gram approach. It is called so because the best tag for a given word is determined by the probability at which it occurs with the n previous tags.

Approaches to Stochastic Tagging: Word Frequency Approach

- In this approach, stochastic taggers disambiguate words based on the probability that a word occurs with a particular tag.
- The tag encountered most frequently with the word in the training set is the one assigned to an ambiguous instance of that word.
- **Example**: If the word "bank" most frequently appears as a noun (NN) in the training data, it will be tagged as NN when encountered in a new context.
- **Issue**: This approach may yield inadmissible sequences of tags. For example, "the bank" and "will bank" would both tag "bank" as NN without considering the context.

Approaches to Stochastic Tagging: Tag Sequence Probabilities (N-gram Approach)

- This approach calculates the probability of a given sequence of tags occurring.
- The best tag for a given word is determined by the probability at which it occurs with the preceding n tags.
- Example: Using bigram (2-gram) probabilities, the word "bank" following "the" is likely to be a noun (NN), whereas "bank" following "will" is likely to be a verb (VB).
- Advantage: This method better captures context and reduces the chances of inadmissible tag sequences.

Combining Approaches for Improved Tagging

- To improve the accuracy and efficiency of POS tagging, a combination of these stochastic approaches is often used:
- **Baseline Tagging**: Start with the Word Frequency Approach to assign the most frequent tags.
- **Contextual Adjustment**: Apply the Tag Sequence Probabilities to adjust tags based on context, using bigram or trigram models to ensure sequences are probable and contextually appropriate.

Example Combined Approach

- **Sentence**: "He can fish."
- **Baseline Tagging (Word Frequency Approach):**
- He (PRP)
- can (MD - most frequent tag for "can")
- fish (NN - most frequent tag for "fish")
- **Contextual Adjustment (N-gram Approach):**
- Based on the context, adjust tags where needed.
- "can fish" might be adjusted from MD NN to VB VB if "can" is frequently a verb when followed by another verb.
- He (PRP)
- can (VB - adjusted based on context)
- fish (VB)

Transformation-based tagging (TBT)

- Transformation-based tagging (TBT) is a method of part-of-speech (POS) tagging that uses a series of rules to transform the tags of words in a text.
- This is in contrast to rule-based POS tagging, which assigns tags to words based on pre-defined rules, and to statistical POS tagging, which relies on a trained model to predict tags based on probability.

Transformation-based tagging (TBT): Working Principles

- Here is an example of how a TBT system might work:
- **Sentence:** "The run was long."
  - Original tags: The (Determiner), run (Verb), was (Verb), long (Adjective)

1. Define a set of rules for transforming the tags of words in the text. For example:

- Rule 1: If the word is a verb and appears after a determiner, change the tag to “noun.”
- Rule 2: If the word is a noun and appears after an adjective, change the tag to “adjective.”

2. Iterate through the words in the text and apply the rules in a specific order. For example:

- A sentence where a verb might appear after a determiner and be transformed:
  - Applying the rule: "run" after "the" could be seen as a noun.
  - Transformed tags: The (Determiner), run (Noun), was (Verb), long (Adjective)
- In this case, the verb "run" is contextually a noun (the act of running), so changing its tag to "noun" is appropriate.

3. Output the transformed tags for each word in the text.

Hidden Markov Model POS tagging

- Hidden Markov models (HMMs) are a type of statistical model that can be used for part-of-speech (POS) tagging in natural language processing (NLP).
- In an HMM-based POS tagger, a model is trained on a large annotated corpus of text to learn the patterns and characteristics of different parts of speech.
- The model uses this training data to predict the POS tag of a given word based on the probability of different tags occurring in the context of the word.

Corpus and Annotated Corpus

- A corpus (plural: corpora) is a large and structured set of texts. It is used in linguistics and natural language processing (NLP) for various tasks such as language analysis, machine learning, and statistical modeling.
- A corpus can include any kind of text, such as books, articles, websites, transcriptions of spoken language, etc.
- **Examples:**
- A collection of all the novels by a particular author.
- A database of newspaper articles.
- Transcripts of recorded conversations.
- Corpora provide the raw data needed for linguistic research and for training language models in NLP tasks.

Annotated Corpus

- An annotated corpus is a corpus that has been enriched with additional information.
- This additional information, or annotations, typically includes linguistic data such as part-of-speech (POS) tags, syntactic structures, semantic roles, and more. Annotations are crucial for training supervised machine learning models because they provide the necessary labels or ground truth.
- **Examples of Annotations:**
- POS Tags: Labels that indicate the part of speech of each word (e.g., noun, verb, adjective).
- Syntactic Structures: Tree structures that represent the grammatical relationships between words in a sentence.
- Named Entities: Labels that identify proper names, such as names of people, organizations, and locations.
- Semantic Roles: Information about the roles that words play in a sentence, such as who did what to whom.

Example of an Annotated Sentence

- Raw sentence:
  - "The quick brown fox jumps over the lazy dog."
- Annotated with POS tags: "The/DT quick/JJ brown/JJ fox/NN jumps/VBZ over/IN the/DT lazy/JJ dog/NN."
- In this annotated sentence:
- "The" is tagged as a determiner (DT).
- "quick" and "brown" are tagged as adjectives (JJ).
- "fox" and "dog" are tagged as nouns (NN).
- "jumps" is tagged as a verb in the third person singular present (VBZ).
- "over" is tagged as a preposition (IN).

Example Annotated Corpus

- "A/DT dog/NN barks/VB."
- "The/DT cat/NN sleeps/VB."
- "A/DT horse/NN runs/VB."
- "The/DT dog/NN eats/VB."
- "A/DT bird/NN flies/VB.“
- This has 5 sentences

Machine Learning-Based Tagging

- Machine Learning-Based Tagging:
- Uses machine learning algorithms.
- Models trained on annotated corpora to predict tags.
- Examples: Support Vector Machines (SVM), Conditional Random Fields (CRF), Neural Networks.
- Deep Learning-Based Tagging:
- Utilizes deep neural networks.
- Can automatically learn features from data.
- Examples: Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Transformer models.

PoS Tagging Techniques: Summary

- Rule-Based Tagging: Good for languages with rich morphology but requires expert knowledge.
- Statistical Tagging: Balances accuracy and computational efficiency, widely used.
- Transformation-Based Tagging: Hybrid approach, effective but can be complex.
- Machine Learning-Based Tagging: Provides good accuracy with sufficient training data.
- Deep Learning-Based Tagging: High accuracy, especially useful with large datasets, but computationally intensive.

Challenges in POS Tagging

- **Ambiguity:** Some words can have multiple POS tags depending on the context in which they appear, making it difficult to determine their correct tag. For example, the word “bass” can be a noun (a type of fish) or an adjective (having a low frequency or pitch).

- **Out-of-vocabulary (OOV) words:** Words that are not present in the training data of a POS tagger can be difficult to tag accurately, especially if they are rare or specific to a particular domain.

- **Complex grammatical structures:** Languages with complex grammatical structures, such as languages with many inflections or free word order, can be more challenging to tag accurately.

- **Lack of annotated training data:** Some languages or domains may have limited annotated training data, making it difficult to train a high-performing POS tagger.

- **Inconsistencies in annotated data:** Annotated data can sometimes contain errors or inconsistencies, which can negatively impact the performance of a POS tagger.

Natural Language Processing (NLP) - Various approaches

- Natural Language Processing (NLP) encompasses various approaches to analyze, understand, and generate human language.
- These approaches have evolved significantly over time and can be broadly categorized into
  - rule-based,
  - statistical, and
  - machine learning (ML) methods.
- Each approach has its own strengths, weaknesses, and appropriate use cases.

1. Rule-based Approaches

- Rule-based systems rely on handcrafted linguistic rules created by human experts.
- These rules are based on grammatical, syntactic, and semantic patterns in the language.
- Strengths
  - Precision: High precision for specific tasks where rules can be accurately defined.
  - Transparency: Easy to understand and debug, as the rules are explicitly defined.
  - Control: Full control over the system's behavior, making it predictable and consistent.

1. Rule-based Approaches

- Weaknesses
  - Scalability: Difficult to scale and maintain for large and complex datasets due to the extensive manual effort required.
  - Adaptability: Limited ability to adapt to new domains or languages without significant rework.
  - Coverage: Often struggles with coverage, missing many linguistic variations not captured by the rules.
- Use Cases
  - Early spell checkers and grammar checkers.
  - Simple information extraction tasks.
  - Specific domain applications where language patterns are well-defined.

2. Statistical Approaches

- Statistical methods leverage the probabilistic analysis of large language corpora.
- These methods use statistical models to predict the likelihood of linguistic phenomena based on observed data.
- Strengths
  - Data-Driven: Can automatically learn patterns from large datasets without needing explicit rules.
  - Coverage: Better coverage of linguistic variations due to the reliance on actual language usage.
  - Scalability: Easier to scale to larger datasets and different languages.

2. Statistical Approaches

- Weaknesses
  - Resource Intensive: Requires large amounts of annotated data for training and substantial computational resources.
  - Interpretability: Often less interpretable than rule-based approaches, making it difficult to understand how decisions are made.
  - Sparsity Issues: Struggles with rare events or low-frequency terms due to data sparsity.
- Use Cases
  - Part-of-speech tagging using Hidden Markov Models (HMMs).
  - Named entity recognition with Conditional Random Fields (CRFs).
  - Statistical machine translation.

3. Machine Learning Approaches

- Machine learning approaches, particularly those using deep learning, have become the dominant paradigm in NLP.
- These methods involve training neural networks on large datasets to learn representations of language.
- Strengths
  - Performance: Often achieves state-of-the-art performance on a wide range of NLP tasks.
  - Flexibility: Can be applied to a variety of tasks without needing task-specific modifications.
  - Generalization: Capable of learning complex patterns and generalizing from examples.

3. Machine Learning Approaches …

- Weaknesses
  - Data and Computation: Requires massive datasets and high computational power, particularly for deep learning models.
  - Interpretability: Models, especially deep neural networks, act as black boxes, making it hard to understand how decisions are made.
  - Resource Dependency: Dependent on high-quality labeled data and extensive preprocessing.
- Use Cases
  - Sentiment analysis with recurrent neural networks (RNNs) or transformers.
  - Machine translation using transformer-based models like BERT and GPT.
  - Question answering systems.

Comparison of approaches

- Rule-Based vs. Statistical
  - Rule-based systems excel in precision for well-defined tasks but struggle with scalability.
  - Statistical methods offer better scalability and coverage but require large datasets and are less interpretable.
- Statistical vs. Machine Learning
  - While statistical models provide a solid foundation and are more interpretable, machine learning approaches, especially deep learning, significantly outperform them in complex tasks due to their ability to capture intricate patterns in data.
- Hybrid Approaches
  - Often, a hybrid approach combining rule-based and statistical/ML methods is used to leverage the strengths of both.
  - For instance, rule-based preprocessing can improve the quality of data fed into machine learning models.

Comparison of approaches …

- The choice of approach depends on the specific requirements of the NLP task at hand, including the availability of data, the need for interpretability, and the computational resources.
- As the field continues to evolve, machine learning and particularly deep learning models are leading the way, but rule-based and statistical methods still play a crucial role in specific scenarios.

MCQs

1) What is the main challenges of NLP?

A. Handling Tokenization B. Handling POS-Tagging

C. Handling Ambiguity of Sentences D. None of the above

2) All of the following are challenges associated with natural language processing except

A. **Dividing up a text into individual words in English.** B. understanding the context in which something is said.

C. recognizing typographical or grammatical errors in texts D. distinguishing between words that have more than one meaning

3) In linguistic morphology, _____ is the process for reducing inflected words to their root form.

A. Stemming B. Rooting

C. Text-Proofing D.Both A and B

4) Morphological Segmentation

A. Is an extension of propositional logic

B. Does Discourse Analysis

C. **Separate words into individual morphemes and identify the class of the morphemes**

D. None of the mentioned

N-grams

- N-grams are contiguous sequences of n items from a given sample of text or speech.
- These items can be
  - phonemes,
  - syllables,
  - letters,
  - words, or
  - base pairs according to the application.
- N-grams are widely used in various natural language processing (NLP) tasks because they provide a simple yet effective way to model and analyze language.
- Here is an overview of N-grams, their types, applications, and challenges.

N-grams

- N-grams are a fundamental concept in NLP, providing a straightforward way to model language and capture local dependencies.
- Despite their simplicity, N-grams face challenges like data sparsity and limited context representation.
- Advanced techniques and models, such as neural networks and transformers, have built upon the basic idea of N-grams to provide more sophisticated and context-aware language processing capabilities.

Types of N-grams

1.Unigrams (1-grams):

1. Single items (words) in a sequence.
2. Example: For the sentence "The cat sat on the mat," the unigrams are "The," "cat," "sat," "on," "the," "mat."

2.Bigrams (2-grams):

1. Pairs of consecutive items.
2. Example: "The cat," "cat sat," "sat on," "on the," "the mat."

3.Trigrams (3-grams):

1. Triples of consecutive items.
2. Example: "The cat sat," "cat sat on," "sat on the," "on the mat."

4.Higher-order N-grams:

1. Sequences of four or more consecutive items (quadgrams, pentagrams, etc.).
2. Example (4-gram): "The cat sat on," "cat sat on the," "sat on the mat."

Applications of N-grams

1. **Language Modeling**
    1. Predicting the next word in a sequence based on the preceding n−1n-1n−1 words.
    2. Used in applications like predictive text input and speech recognition.
2. **Text Generation**
    1. Generating text by predicting the next word or phrase in a sequence.
    2. Used in chatbots and automated content creation.
3. **Spell Checking and Correction**
    1. Identifying and correcting misspelled words based on common word sequences.
4. **Machine Translation**
    1. Translating text from one language to another by modeling language patterns.
5. **Information Retrieval**
    1. Enhancing search engines by understanding and matching user queries with relevant documents.
6. **Sentiment Analysis**
    1. Analyzing sequences of words to determine the sentiment expressed in a text.
7. **Text Classification**
    1. Classifying documents or sentences into categories based on the frequency of N-grams.

Challenges with N-grams

1.Data Sparsity

1. As n increases, the number of possible N-grams grows exponentially, leading to many rare or unseen N-grams in the training data.

2.Context Limitation

1. Fixed-size N-grams may not capture long-range dependencies in text, limiting their ability to model context effectively.

3.Dimensionality

1. Higher-order N-grams result in high-dimensional feature spaces, which can be computationally expensive and challenging to manage.

4.Smoothing

1. Addressing zero probabilities for unseen N-grams requires smoothing techniques like Laplace smoothing, Good-Turing discounting, or back-off models.

Smoothing Techniques

There are several techniques used in natural language processing for handling the issue of zero probabilities in n-gram models.

1. Laplace Smoothing (Add-One Smoothing)
2. Good-Turing Discounting
3. Back-Off Models
4. Kneser-Ney Smoothing

Here is a brief explanation and use case for each smoothing technique:

Smoothing Techniques …

- **Laplace Smoothing (Add-One Smoothing)**
- Adds one to the count of each N-gram to ensure no zero probabilities.
- Formula:
  - P(wi | wi-n+1 to wi-1) = (C(wi-n+1 to wi-1) + 1) / (V + |W|)
  - P(wi | wi-n+1 to wi-1): This represents the smoothed probability of word wi appearing after the context (sequence of words) wi-n+1 to wi-1 (an N-gram).
  - C(wi-n+1 to wi-1): This represents the count of the context (sequence of words) wi-n+1 to wi-1 in the training data.
  - V: This represents the total size of the vocabulary (the number of unique words).
  - |W|: This represents the size of the N-gram (the number of words in the context sequence).
  - 1: This is the added one for smoothing.

- **Good-Turing Discounting**
- This technique refines Laplace Smoothing by considering the frequency of N-grams with specific counts.
- It adjusts the probabilities based on the idea that N-grams with a count of one are more likely to actually have a higher probability than what Laplace Smoothing assigns.

Smoothing Techniques …

•**Back-Off Models**
•Use lower-order N-grams when higher-order N-grams have zero counts.
•Formula:

- **P(wi | wi-n+2 to wi-1) = β(wi-n+1 to wi-1) * P(wi | wi-n+2 to wi-1)**
- where:
- P(wi | wi-n+2 to wi-1): The probability of word wi appearing after the context (sequence of words) wi-n+2 to wi-1 (notice the context length is n-1).
- β(wi-n+1 to wi-1): The backoff weight for the context wi-n+1 to wi-1. This weight determines how much probability mass is "backed off" from the unseen N-gram and redistributed to lower-order N-grams.
- P(wi | wi-n+2 to wi-1): The probability of word wi appearing after the shorter context wi-n+2 to wi-1 (a lower-order N-gram).

- This formula essentially says:
- If the N-gram with context (wi-n+1 to wi-1) is unseen (zero count), we "back off" and use a weight (β) to redistribute some probability mass from the unseen N-gram.
- This weight (β) is then multiplied by the probability of the word wi appearing after the shorter context (wi-n+2 to wi-1).
- There are different variations of Back-Off Models, and the specific formula might differ slightly depending on the implementation. However, the core idea of using lower-order N-grams as backups and redistributing probability mass using backoff weights remains consistent.

- **Kneser-Ney Smoothing**
- Considers the diversity of contexts in which words appear, providing better performance for rare words.
- This is a more advanced technique that considers the variety of contexts in which words appear.
- It provides better performance for rare words by taking into account the richness of the context they appear in.

MCQs

1.N-grams are defined as the combination of N keywords together. How many bi-grams can be generated from the given sentence: ***The Father of our nation is Mahatma Gandhiji***

A.8 B.9

C.7 D.4

2. It is the development of probabilistic models that can predict the next word in the sequence given the words that precede.

A. **Statistical Language Modelling** B. Probabilistic Langualge Modeling

C. Neural Language Modelling D. Natural Language Understanding

3. It is a measure of how good a probability distribution predicts a sample

A. Entropy B. **Perplexity**

C. Cross-Entropy D. Information Gain

4.What are the python libraries used in NLP?

A. Pandas B. NLTK

C. Spacy D. **All the mentioned above**

Multiword expressions (MWEs)

- Multiword expressions (MWEs) refer to sequences of words that have a collective meaning that is not necessarily predictable from the meanings of the individual words.
- In natural language processing (NLP), MWEs pose challenges because their meaning may not be transparent based on the meanings of the constituent words alone.
- Instead, their meaning often arises from idiomatic usage or conventional linguistic patterns.

Multiword expressions (MWEs)

- Sequence of Words: MWEs are sequences of words that are not necessarily contiguous in a concrete utterance. They do not always appear in the same order in each utterance.
- Ambiguity between Type and Token: There is intentional ambiguity between the type (general form) and token (specific instance) of MWEs. This includes the distinction between inflected word forms and their lemmas.
- Ambiguity in Interpretation: MWEs present ambiguity between:
- Narrow interpretation: Character sequences separated from other character sequences by spaces and other separators.
- Broad interpretation: Abstract lexical units of the grammar.
- Non-Compounded Properties: MWEs possess properties that are not predictable from the individual components and their normal mode of combination.
- These sentences describe MWEs in terms of their structure, interpretation, and characteristics, addressing the potential ambiguities and unique properties of such expressions.

MWEs: Forms

- Idioms: Phrases where the meaning is not directly derived from the literal interpretation of the words, such as "kick the bucket" or "bite the bullet."
- Phrasal verbs: Verbs combined with one or more particles (prepositions or adverbs) to convey a distinct meaning, such as "take off" or "look up."
- Compound nouns: Noun phrases composed of multiple words that function together as a single unit, like "ice cream" or "toothpaste."
- Named entities: Multiword expressions that refer to specific entities, such as "New York City" or "United States of America."
- Fixed expressions: Common phrases or collocations where the words are frequently used together with a specific meaning, like "by and large" or "more or less."

Dealing with MWEs in NLP tasks

- Dealing with MWEs in NLP tasks like parsing, machine translation, sentiment analysis, and information retrieval requires specialized handling.
- Approaches include:
  - MWE Identification
  - MWE Disambiguation
  - MWE Representation
  - MWE Integration

Dealing with MWEs in NLP tasks …

- MWE Identification
  - Recognizing MWEs within text is essential for accurate processing.
  - This can involve rule-based methods, statistical approaches, or a combination of both.
- MWE Representation
  - Deciding how to represent MWEs in NLP models is crucial.
  - Some approaches treat them as single tokens, while others decompose them into their constituent parts.

Dealing with MWEs in NLP tasks …

- MWE Disambiguation
  - Resolving ambiguity in the interpretation of MWEs is often necessary.
  - Contextual cues or specific disambiguation algorithms may be employed.
- MWE Integration
  - Incorporating MWE information into various NLP tasks can enhance performance.
  - For example, recognizing and preserving idiomatic expressions during machine translation can improve translation quality.
- Addressing MWEs effectively is vital for advancing the accuracy and naturalness of NLP systems, as they play a significant role in human communication and understanding.

Addressing MWEs in NLP Tasks

- Effectively handling MWEs is crucial for various NLP tasks, including:
- **Parsing:** Breaking down sentences into grammatical structures. MWEs often require specialized parsing strategies.
- **Machine Translation:** Translating text from one language to another while preserving the meaning of MWEs, especially idioms.
- **Sentiment Analysis:** Understanding the emotional tone of text. MWEs can influence sentiment, like "spill the tea" (gossip) having a negative connotation.
- **Information Retrieval:** Finding relevant information in text. Recognizing MWEs can improve search accuracy.

MWETokenizer

- The multi-word expression tokenizer is a rule-based, “add-on” tokenizer offered by NLTK. Once the text has been tokenized by a tokenizer of choice, some tokens can be re-grouped into multi-word expressions.
- For example, the name Martha Jones is combined into a single token instead of being broken into two tokens. This tokenizer is very flexible since it is agnostic of the base tokenizer that was used to generate the tokens.

```
tokenizer = MWETokenizer()
tokenizer.add_mwe(('Martha', 'Jones'))
print(f'Multi-word expression (MWE) tokenization = {tokenizer.tokenize(word_tokenize(sentence))}')
```

```
Multi-word expression (MWE) tokenization = ['It', "'s", 'true', ',', 'Ms.', 'Martha_Jones', '!', '#', 'Truth']
```

Collocations

- Collocations are phrases or expressions containing multiple words, that are highly likely to co-occur.
- For example - ‘social media’, ‘school holiday’, ‘machine learning’, ‘Universal Studios Singapore’, etc.
- Collocations are combinations of words that frequently occur together and exhibit a strong association, such as "strong tea" or "make a decision."
- Identifying collocations is essential for various NLP tasks, including language modeling, machine translation, and information retrieval.
- More examples: verb-noun pairs ("make a decision"), adjective-noun pairs ("strong tea"), or phrasal verbs ("run out of").

Collocations (Association Measures, Coefficients and Context Measures)

- Several statistical measures and techniques are used to identify and evaluate collocations.
- These can be broadly categorized into association measures, coefficients, and context measures.

Collocations (Association Measures)

- Association measures evaluate the strength of the relationship between words in a potential collocation.
- Common measures include:
    1. Mutual Information (MI)
    2. Pointwise Mutual Information (PMI)
    3. Chi-Square Test
    4. Likelihood Ratio
    5. Dice Coefficient

Collocations (Association Measures) …

1.Mutual Information (MI)

1. Measures how much information the occurrence of one word provides about the occurrence of another word.
2. Formula: MI(x,y)= MI(x, y) = log (P(x, y) / (P(x) * P(y)))

   P(x, y): Joint probability of x and y occurring together.

   P(x): Probability of word x occurring.

   P(y): Probability of word y occurring.
3. Higher MI indicates a stronger association, but it can be biased towards low-frequency words.

Collocations (Association Measures) …

2. Pointwise Mutual Information (PMI)

- A variant of MI that considers individual occurrences.
- Formula: $PMI(x,y)=\log (P(x, y) / (P(x) * P(y)))$
- (Same formula as MI)
- Like MI, it is prone to favoring rare word pairs.

Collocations (Association Measures) …

3. Chi-Square Test

- Evaluates the independence of words in a collocation.
- Formula: $\chi 2$ (Chi-square)$=\sum$ (Oi−Ei)2 / Ei where Oi is the observed frequency and Ei is the expected frequency.
- (Calculated based on individual word probabilities)
- Effective for high-frequency words but can be influenced by large sample sizes.

Collocations (Association Measures) …

4. Likelihood Ratio

•Compares the likelihood of the data under two hypotheses: words occurring independently versus co-occurring. (i.e., Words are independent. Words co-occur more than expected by chance)

•Useful for distinguishing genuine collocations from random co-occurrences.

•A high likelihood ratio suggests a genuine collocation

•(Formula for Likelihood Ratio can vary depending on implementation. Here's a general form)

•Likelihood Ratio = (Likelihood of data assuming co-occurrence) / (Likelihood of data assuming independence)

Collocations (Association Measures) …

5. Dice Coefficient

- Measures the similarity between two sets (the words in the collocation). This measure focuses on the overlap between the sets of documents where words x and y appear.
- Formula: $Dice(x,y) = 2 \times C(x,y) / C(x)+C(y)$
  - C(x, y): Number of documents where both words x and y appear.
  - C(x): Number of documents where word x appears.
  - C(y): Number of documents where word y appears.
- Balances frequency and co-occurrence effectively.

Collocations (Coefficients)

- Coefficients are specific types of association measures designed to quantify the strength of the relationship between words.
- Some of the coefficients are:
  1. Jaccard Coefficient
  2. T-score
  3. Phi Coefficient

Collocations (Coefficients) …

1.Jaccard Coefficient

This coefficient measures the overlap between the sets of documents where words x and y appear. It provides a value between 0 and 1, with higher values indicating a stronger co-occurrence relationship.

Formula:

$$\text{Jaccard}(x,y) = C(x,y) / C(x)+C(y)-C(x,y)$$

- C(x, y): Number of documents where both words x and y appear.
- C(x): Number of documents where word x appears.
- C(y): Number of documents where word y appears.

Useful for comparing word pairs' co-occurrence.

Collocations (Coefficients) …

2. T-score

- This coefficient evaluates the statistical significance of the difference between the observed co-occurrence frequency (C(x, y)) and the expected co-occurrence frequency (E(x, y)) under the assumption of independence.
- Higher T-scores indicate a more statistically significant co-occurrence.
- Formula:
- $T(x, y) = (C(x, y) - E(x, y)) / \sqrt{(E(x, y))}$
  - C(x, y): Number of times words x and y co-occur.
  - E(x, y): Expected co-occurrence frequency, typically calculated based on individual word frequencies.

Collocations (Coefficients) …

3. Phi Coefficient

- This coefficient measures the correlation between the binary variables representing the occurrence (1) or non-occurrence (0) of words x and y.
- It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no correlation.
- Formula:
- Φ (phi) = (AD - BC) / $\sqrt{(A+B)(C+D)(A+C)(B+D)}$
  - A: Number of documents where both x and y occur.
  - B: Number of documents where only x occurs (but not y).
  - C: Number of documents where only y occurs (but not x).
  - D: Number of documents where neither x nor y occur.

Collocations (Context Measures)

- Context measures focus on the contextual usage of words to determine collocations.
  1. Window-Based Co-Occurrence
  2. Dependency Parsing:
  3. Latent Semantic Analysis (LSA)
  4. Word Embeddings (e.g., Word2Vec, GloVe)

Collocations (Context Measures)

- **Window-Based Co-Occurrence**
- Counts the number of times words co-occur within a specified window size in a text.
- Larger windows capture broader contextual relationships but may introduce noise.
- **Dependency Parsing**
- Uses syntactic relationships identified by a parser to find collocations.
- More precise than window-based methods but computationally intensive.

Collocations (Context Measures) …

- **Latent Semantic Analysis (LSA)**
- Projects words into a latent space where collocations are identified based on semantic similarity.
- Effective for capturing deeper semantic relationships but requires significant computational resources.
- **Word Embeddings (e.g., Word2Vec, GloVe)**
- Represent words in a continuous vector space where collocations are identified based on vector similarity.
- Highly effective for capturing both syntactic and semantic relationships.

Strong Collocations

- **Strong Collocations: Powering Language Models**
- Strong collocations are word pairs with a high frequency of co-occurrence and a strong statistical association. These are crucial for language models because they help predict the next word accurately. Here are some examples:
- **"Machine Learning":** This collocation has a high Pointwise Mutual Information (PMI) value, indicating that these words frequently appear together in technical texts.
- **"Climate Change":** Both PMI and chi-square test values are likely to be high for "climate change," as these terms are commonly found together in environmental discussions.

Weaker Collocations

- **Weaker Collocations: Broadening Information Retrieval**
- Weaker collocations show a lower frequency of co-occurrence but still hold value in information retrieval systems. They help identify relevant documents by capturing a broader range of related concepts, even if the words don't always appear together as frequently. Here are some examples:
- **"Renewable Energy":** The co-occurrence frequency and Dice coefficient for "renewable energy" might be moderate, but this collocation is useful for broader searches in information retrieval related to sustainability.
- **"Artificial Intelligence":** While "artificial intelligence" might not always co-occur with "renewable energy," they share a connection to technological advancements. This semantic relevance makes it a valuable term for information retrieval systems, even if the co-occurrence count is not as high as a strong collocation.
- By understanding the distinction between strong and weak collocations, we can leverage their strengths for different NLP tasks.

Vector representation of words

- Vector representation of words, also known as word embeddings, is a foundational concept in modern natural language processing (NLP).
- Word embeddings map words or phrases to continuous vector spaces, capturing semantic and syntactic information in a form that can be efficiently processed by machine learning models.
- Here’s an overview of key concepts, methods, and applications related to word embeddings.

Vector representation of words

- Vector representation of words is a critical aspect of NLP, enabling machines to understand and process human language more effectively.
- With the advent of advanced techniques like transformers and contextual embeddings, NLP models have achieved remarkable improvements in various language understanding tasks.
- As research continues, we can expect even more sophisticated methods for capturing the richness of human language in computational forms.

Vector representation of words: Key Concepts

- Dimensionality: The number of dimensions in the vector space, typically ranging from 50 to 300 dimensions for practical applications. Higher dimensions can capture more nuances but require more computational resources.
- Semantic Similarity: Words with similar meanings have vectors that are close together in the embedding space. For example, the vectors for "king" and "queen" are close, as are the vectors for "apple" and "orange."
- Contextual Information: Embeddings capture the context in which words appear, allowing models to differentiate between different meanings of the same word based on surrounding words.

Methods for Generating Word Embeddings

- There are six methods for Methods for Generating Word Embeddings
  1. Word2Vec
  2. GloVe (Global Vectors for Word Representation)
  3. FastText
  4. ELMo (Embeddings from Language Models)
  5. BERT (Bidirectional Encoder Representations from Transformers)
  6. Transformer-based Models (e.g., GPT, T5)

Methods for Generating Word Embeddings …

1. Word2Vec

- Developed by Google, this method includes two main algorithms:
  - Continuous Bag of Words (CBOW): Predicts a target word from its surrounding context words.
  - Skip-gram: Predicts surrounding context words given a target word.
  - Both algorithms learn embeddings by optimizing for the co-occurrence probability of words within a fixed-size window.

2. GloVe (Global Vectors for Word Representation)

- Developed by Stanford, GloVe creates embeddings by factorizing the co-occurrence matrix of words in a corpus.
- It captures global statistical information by considering the ratio of word co-occurrences.

Methods for Generating Word Embeddings …

3. FastText

- An extension of Word2Vec by Facebook, FastText represents words as bags of character n-grams.
- This approach allows the model to generate embeddings for out-of-vocabulary words by composing them from known n-grams, thus handling morphological variations better.

4. ELMo (Embeddings from Language Models)

- Developed by AllenNLP, ELMo generates contextualized embeddings using a deep bidirectional language model.
- Each word's representation is derived from the entire sentence, capturing context-specific meaning.

Methods for Generating Word Embeddings …

5. BERT (Bidirectional Encoder Representations from Transformers)

- Developed by Google, BERT uses transformer architecture to generate deep contextualized embeddings.
- It captures context from both directions (left-to-right and right-to-left) and has set new benchmarks in various NLP tasks.

6. Transformer-based Models (e.g., GPT, T5)

- These models use transformer architectures to create embeddings that are highly contextualized and can be fine-tuned for specific tasks.

Word Embeddings: Applications

- Text Classification: Embeddings are used as input features for models that classify text into categories, such as spam detection or sentiment analysis.
- Machine Translation: Embeddings help in translating text from one language to another by capturing semantic similarities across languages.
- Information Retrieval: Embeddings enhance search engines by improving the understanding of query intent and document relevance.

Word Embeddings: Applications …

- Named Entity Recognition (NER): Embeddings assist in identifying and classifying entities within text, such as names of people, organizations, and locations.
- Question Answering: Embeddings enable models to understand and generate accurate responses to user queries by capturing the context of the question and potential answers.

Advantages of Word Embeddings

- Capturing Semantic Relationships: Word embeddings capture nuanced relationships between words, enabling models to understand language more deeply.
- Dimensionality Reduction: Transforming high-dimensional categorical data (words) into lower-dimensional continuous vectors makes the data more manageable for machine learning algorithms.
- Transfer Learning: Pre-trained embeddings on large corpora can be transferred to various downstream tasks, reducing the need for extensive task-specific training data.

Challenges

- Context Sensitivity: Traditional embeddings like Word2Vec and GloVe are static and do not capture different meanings of words in different contexts.
- Contextual embeddings like BERT address this but require more computational resources.
- Resource Intensive: Training large models like BERT requires significant computational power and large datasets, which may not be feasible for all organizations.
- Interpretability: Understanding what specific dimensions in an embedding vector represent can be challenging, making it harder to interpret model decisions.

Language Models

- A Language Model is a probabilistic model which predicts the probability that a sequence of tokens belongs to a language.
- **How Language Models helps in NLP Tasks**
- The probabilities returned by a language model are mostly useful to compare the likelihood that different sentences are "good sentences".
- Spell checking, Automatic Speech Recognition, Machine Translation:

N-grams

- N-grams are the simplest tool available to construct a language model.
- An N-gram is a sequence of N words.
- An N-gram model predicts the probability of a given N-gram within any sequence of words in the language.

N-grams in Language Models

- N-grams are fundamental building blocks for statistical language models. They represent sequences of N words, where N refers to the number of words in the sequence. Here's a breakdown of commonly used n-grams:
- **Unigram (n=1):** Represents single words. A unigram model estimates the probability of a particular word (wi) appearing in a sequence.
- **Bigram (n=2):** Represents sequences of two words. A bigram model estimates the probability of a word (wi) appearing given the previous word (wi-1) in the sequence. This leverages the concept of a Markov process, where the probability of a word depends only on the preceding word.
- **Trigram (n=3):** Represents sequences of three words. A trigram model estimates the probability of a word (wi) appearing given the two preceding words (wi-1 and wi-2) in the sequence. Trigram models capture slightly more context compared to bigrams.

Estimating Sentence Probability using N-grams

- The chain rule of probability allows us to estimate the probability of an entire sentence (P(s)) by considering the probabilities of individual n-grams in the sequence. Here's the formula breakdown:
- P(s) = P(w1, w2, w3, ..., wn): This represents the probability of the entire sentence, where wi represents each word in the sequence from the first word (w1) to the last word (wn).
- We can rewrite this using the chain rule:
- P(w1w2...wn) = ∏ P(wi | w1w2...wn-1): This breaks down the sentence probability into a product of individual n-gram probabilities. Here, P(wi | w1w2...wn-1) represents the conditional probability of the current word (wi) given the preceding n-1 words (w1w2...wn-1) in the sequence. The product (∏) iterates over all words in the sentence.
- However, directly calculating these conditional probabilities becomes impractical for higher n-grams due to data sparsity issues (limited data for all possible n-gram combinations). In practice, various techniques like smoothing are used to address this challenge.

Unigram Model (n=1)

- **Unigram Model (n=1)**
- **Definition:** A unigram model calculates the probability of each word independently, without considering surrounding context.
- **Equation:** P(wi)
- **Example:** P("apple")

Bigram Model (n=2)

- **Definition:** A bigram model predicts the probability of a word given its preceding word, assuming a Markov process (where the probability of a word depends only on the word before it).
- **Equation:** $P(w_i \mid w_{i-1})$
- **Example:** P("apple" | "the")
- **Chain Rule of Probability:**
- **Equation:** $P(w_1 w_2 \ldots w_n) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1 w_2) \times \ldots \times P(w_n \mid w_{n-1})$

Trigram Model (n=3)

- **Definition:** A trigram model predicts the probability of a word given its two preceding words.
- **Equation:** $P(w_i \mid w_{i-2}, w_{i-1})$
- **Example:** P("apple" | "eat", "I")
- **Chain Rule of Probability:**
- **Equation:** $P(w_1 w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 w_2 \ldots w_{i-1})$

Estimating Probability

- We can estimate the probability of a sequence (sentence) using the chain rule by multiplying the probabilities of individual n-grams in the sequence.
- However, directly calculating these conditional probabilities becomes difficult for higher n-grams due to data sparsity (limited data for all possible n-gram combinations).
- In practice, various techniques like smoothing are used to address this challenge.

Problem 1:

Find the probability of test sentence given below if a bigram model is used on the following training sentences

Training data:

- The Arabian Knights
- These are the fairy tales of the east
- The Stories of the Arabian Knights are translated primary languages.

Test Data:

- The Arabian Knights are the fairy tales of the east

Bigram Model Solution

- To compute a particular bigram probability of a word wn given a previous word wn−1, we'll compute the count of the bigram C(wn−1wn) and normalize by the sum of all the bigrams that share the same first word wn−1:
  - Estimating bigram probabilities
  - The Maximum Likelihood Estimate
  - $$\mathbf{P(wn \mid wn-1) = \frac{c(wn-1wn)}{c(wn-1)}}$$

The Maximum Likelihood Estimate

- Equation for estimating bigram probabilities using Maximum Likelihood Estimate (MLE)
  - $$P(w_n \mid w_{n-1}) = \frac{c(w_{n-1}w_n)}{c(w_{n-1})}$$
- P(wn | wn-1): This represents the probability of the current word (wn) appearing given the previous word (wn-1) in the sequence.
- c(wn-1wn): This represents the count of how many times the bigram (wn-1 followed by wn) appears in the training data.
- c(wn-1): This represents the count of how many times the previous word (wn-1) appears in the training data (including all bigrams where it's the first word).
- Therefore, the equation essentially calculates the proportion of times the current word (wn) follows the previous word (wn-1) compared to the total occurrences of the previous word (wn-1). This gives us an estimate of how likely wn is to appear after wn-1 in unseen text.

1.The Arabian Knights	2.These are the fairy tales of the east	3. The Stories of the Arabian Knights are translated primary languages.
P(The|<s>)=2/3=0.66 **P(Arabian|The)=2/5=0.4** **P(Knights|Arabian)=2/2=1**	P(these|<s>)=1/3=0.333 P(are|these)=1/1=1 P(the|are)=1/2=0.5 P(fairy|the)=1/5=0.2 P(tales|Fairy)=1/1=1 P(of|tales)=1/1=1 P(the|of)=1/1=1 P(east|the)=1/5=0.2 Note: **P(wn | wn−1) =c(wn−1wn) / c(wn−1)**	P(the|<s>)=2/3=0.66 P(Stories|the)=1/5=0.2 P(of|Stories)=1/1=1 P(the|of)=1/1=1 P(Arabian|The)=2/5=0.4 P(Knights|Arabian)=2/2=1 P(are|Knights)=1/2=0.5 P(translated|are)=1/2=0.5 P(Primary|translated)=1/1=1 P(languages|Primary)=1/1=1

Test Solution

The Arabian Knights are the fairy tales of the east

=P(the|<s>)* P(Arabian| The)* P(Knight | Arabian)* P(are|Knights)* P(the|are)* P(fairy|the)* P(tales|Fairy)* P(of|tales)* P(the|of)* P(east|the)

=0.66*0.4*1*0.5*0.5*0.2*1*1*1*0.2

=0.00268

Problem 2:

- Let's work through an example using a mini-corpus of three sentences.
- We'll first need to augment each sentence with a special symbol <s> at the beginning of the sentence, to give us the bigram context of the first word.
- We'll also need a special end-symbol. </s>
  - <s> I am Sam </s>
  - <s> Sam I am </s>
  - <s> I am not Sam </s>

Training set

- I am Sam
- Sam I am
- I am not Sam

Testing set

- I am not Sam

Solution

Training set

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I am not Sam </s>

Testing set

- I am not Sam

- Note: $P(wn \mid wn-1) = \frac{c(wn-1wn)}{c(wn-1)}$

- **Bi-gram**
  - =P(I/<s>)*P(am/I)*P(not/am)*P(Sam/not)
  - =c(<s>I)/c(<s>) * c(I am)/c(I) * c(am not)/c(am) * c(not Sam)/c<not>
  - =(2/3)*(3/3)*(1/3)*(1/1)
  - =0.221
- **Tri-gram**
  - =P(I/<s1><s2>)*P(am/I<s1>)*P(not/I am)*P(Sam/am not)
  - =(2/3)*(2/2)*(1/3)*(1/1)
  - 0.2211

The probability of each word in Unigram model

- In a unigram model, the probability of each word depends on its own probability in the corpus.
- $P(w_n)=\frac{Number\ \ of\ times\ Wn\ appears}{Total\ number\ of\ words\ in\ the\ corpus}$
- If total number of words in the corpus is equal to 10000 and ‘the’ appears 7000 times, then prob(the)=7000/10000=0.7=70%

Data Sparseness Issue in N gram model

‘zero probability’ one n gram cause a zero probability of the entire sentence. It can be avoided by “Smoothing *n*-gram models”.