VIDYASAGAR COLLEGE OF ARTS AND SCIENCE

UDUMALPET

DEPARTMENT OF DATA SCIENCE

ACADEMIC YEAR : 2025-2026 [EVEN SEMESTER]

CLASS: III B.Sc [DATA SCIENCE]

SUBJECT: NATURAL LANGUAGE PROCESSING

Unit:1 Introduction 10 hours

Natural Language Processing tasks in syntax, semantics, and pragmatics – Issues - Applications - The role of machine learning - Probability Basics –Information theory – Collocations -N-gram Language Models - Estimating parameters and smoothing - Evaluating language models.

Natural Language Processing(NLP):

NLP is a field of artificial intelligence that deals with the interaction between computers and humans in natural language. It's a way machines can understand, interpret, and generate human language.

Natural Language Processing Tasks in Syntax

1. **Part-of-Speech (POS) Tagging**: Identify word types
   - Example: "The cat sleeps." (The/DT, cat/NN, sleeps/VB)
     - DT: Determiner (e.g., "The", "A")
       NN: Noun (e.g., "cat", "dog")
       VB: Verb (e.g., "sleeps", "runs")
     - These are part-of-speech (POS) tags used to categorize words based on their grammatical context.

2. **Named Entity Recognition (NER)**: Identify entities

   Example: "John lives in New York." (John/PERSON, New York/LOCATION)

3. **Dependency Parsing**: Analyze sentence structure

   Example: "The cat sleeps." (cat ← nsubj ← sleeps)
   - "cat" is the nsubj (nominal subject) of "sleeps"
     - The arrow direction shows the dependency relationship:
   - "sleeps" is the head (main verb)
     - "cat" is the dependent (subject of the verb)
   - In simple terms, "cat" is doing the action of "sleeps"

4. **Syntax Tree Construction**: Build tree representations
   Example: "(S (NP The cat) (VP sleeps))"
   - S: Sentence
   - NP: Noun Phrase ("The cat")
   - VP: Verb Phrase ("sleeps")
   - The tree structure shows:
     - "The cat" is a Noun Phrase (NP)
     - "sleeps" is a Verb Phrase (VP)
   - Together, they form a Sentence (S)
   - This represents the hierarchical structure of the sentence

5. **Constituency Parsing**: Break down sentences into phrases
   Example: "The cat sleeps." (NP: The cat, VP: sleeps)

   - The sentence "The cat sleeps" is broken down into:
     - NP (Noun Phrase): "The cat"
     - VP (Verb Phrase): "sleeps"
   - This shows the sentence is composed of a Noun Phrase ("The cat") and a Verb Phrase ("sleeps")
   - Constituency Parsing identifies these phrase groups to understand sentence structure

These tasks help machines understand sentence structure and meaning.

Natural Language Processing Tasks in Semantics

Semantics deals with meaning in language. Key tasks include:

1. Semantic Role Labeling (SRL): Identify roles (e.g., "who", "what", "where")
   Example: "John gave Mary a book."
   John (GIVER), Mary (RECIPIENT), a book (THING GIVEN)

   Sentence: "The chef cooked the meal in the kitchen."

   Roles:
   * Agent (Who): The chef
   * Action (What): cooked
   * Patient (What was acted upon): the meal
   * Location (Where): in the kitchen

   Diagram:
   The chef cooked the meal in the kitchen
   [Agent] [Action] [Patient] [Location]

2. Entity Disambiguation: Resolve word meanings
   Example: "bank" (financial institution vs. riverbank)

   Sentence: "I went to the bank to deposit money."
   Meaning:
   * bank (financial institution)

   Sentence: "The bank of the river was lined with trees."
   Meaning:
   * bank (riverbank)

   Diagram:
   bank (financial institution) ≠ bank (riverbank)
   This shows how the same word can have different meanings based on context

3. Semantic Similarity: Measure meaning similarity
   Example: "car" and "vehicle" are similar
   "car" and "vehicle" are similar because they share related meanings

   • Hypernym: "vehicle" is a broader category (includes cars, bikes, trucks)
   • Synonyms: "car" and "automobile" are very similar
   • Context: Both "car" and "vehicle" can be used in similar contexts
   (e.g., Transportation)
   Example:
   car → vehicle → transportation
   This shows "car" is a type of "vehicle", and both relate to transportation

4. Textual Entailment: Infer meaning from text
   Example: "The cat is on the mat." entails "There is a cat."
   The cat is on the mat." → "There is a cat."
   The first sentence entails (implies) the second sentence .
   If the first sentence is true, the second sentence must also be true .
   The cat is on the mat.
   → There is a cat.
   → Something is on the mat.
   Another Example:
   "John bought a book." → "John owns a book."
   If John bought a book, it's likely he now owns it
   This helps machines understand implied meanings

Natural Language Processing Tasks in Pragmatics

Pragmatics deals with context and implied meaning . Key tasks include:

1. Speech Act Recognition
   Identify the intention behind an utterance:
   Example: "Can you pass the salt?"

Literal meaning: Asking about ability
Intended meaning: Request to pass the salt
Types: requests, apologies, orders, promises, etc.

2. Implicature
   Infer implied meaning:
   Example: "It's cold in here."
   Implied meaning: "Please turn on the heater" or "Let's close the window"
   Types: conversational implicature (context-dependent), conventional implicature (cultural)
3. Deixis Resolution
   Resolve context-dependent words:
   Examples: "here", "now", "he", "this"
   Example: "I'll meet you here tomorrow."
   "here" depends on the speaker's location
4. Discourse Analysis
   Analyze conversation structure and flow:
   Example: Turn-taking, topic shifts, coherence
   Helps understand how conversations work
5. Presupposition
   Identify assumptions behind a statement:
   Example: "John stopped smoking."
   Presupposition: "John used to smoke"
   Helps understand implicit meaning

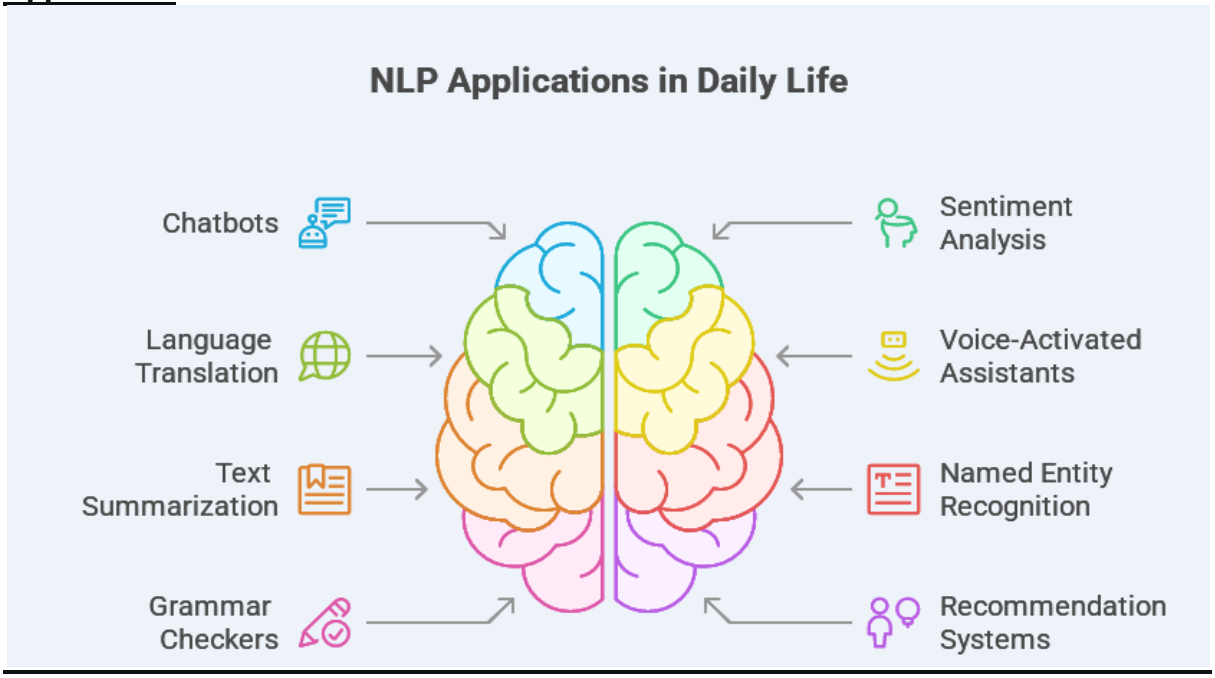
These tasks help machines understand language in context, beyond just literal meaning

Issues in NLP

1. **Ambiguity**: Multiple meanings (e.g., "bank" - financial institution vs. riverbank)
   Example: "I went to the bank." (which bank?)
2. **Contextual Understanding**: Capturing nuances
   Example: "It's cold in here." (request to turn on heater or close window?)
3. **Data Quality**: Noisy, biased data
   Example: Trained on biased reviews, NLP model learns stereotypes
4. **Language Variations**: Dialects, idioms
   Example: "Break a leg" (good luck, not literal)
5. **Common Sense**: Lack of world knowledge
   Example: "The library is open." (implies people can borrow books)

These issues affect NLP system performance

Applications



- **Sentiment Analysis**: Analyze customer reviews, e.g., "I love this product!" → Positive sentiment
- **Chatbots**: Virtual assistants, e.g., Siri, Alexa, Google Assistant
- **Language Translation**: Translate text, e.g., Google Translate (English to Spanish: "Hello" → "Hola")
- **Text Summarization**: Summarize articles, e.g., Summarize a news article into a short paragraph
- **Named Entity Recognition (NER)**: Identify entities, e.g., "John Smith" (Person), "New York" (Location)
- **Speech Recognition**: Transcribe spoken language, e.g., Voice-to-text systems
- **Part-of-Speech (POS) Tagging**: Identify word types, e.g., "The dog runs" → The (Det), dog (Noun), runs (Verb)
- **Question Answering**: Answer questions, e.g., "What is the capital of France?" → Paris

The role of Machine Learning

Machine learning is a subset of artificial intelligence (AI) that involves training algorithms to learn from data and improve their performance on a task over time.

The detailed explanation of the role of machine learning:

1. Predictive Modeling: Machine learning algorithms can be trained to predict outcomes based on historical data, such as predicting customer churn or identifying potential credit risks.

2. Classification: ML models can classify data into categories, like spam vs. non-spam emails, or medical diagnosis (e.g., cancer vs. non-cancer).

3. Clustering: Unsupervised learning techniques group similar data points into clusters, helping identify patterns or customer segments.
4. Recommendation Systems: ML powers recommendation engines, suggesting products or content based on user behavior and preferences (e.g., Netflix, Amazon).
5. Natural Language Processing (NLP): Machine learning is used in NLP tasks like language translation, sentiment analysis, and text summarization.
6. Image and Speech Recognition: ML models can recognize images (e.g., self-driving cars, facial recognition) and speech (e.g., virtual assistants like Siri, Alexa).
7. Decision Making: Machine learning can automate decision-making processes, like credit approval or loan applications, based on data-driven insights.
8. Continuous Improvement: ML models can learn from new data, adapting to changes and improving their performance over time, making them a valuable asset in many industries.

Probability Basics

In machine learning, **probability** refers to a measure of the likelihood of an event or outcome occurring. It's a numerical value between 0 and 1 that represents the chance or probability of an event happening.

In ML, probability is used to:

- **Model uncertainty**: Represent uncertainty in data, models, or predictions.
- **Make predictions**: Predict outcomes or classify instances based on probability distributions.
- **Estimate parameters**: Estimate model parameters using probability-based methods (e.g., maximum likelihood estimation).

Common probability concepts in ML include:

- **Probability distribution**: A function describing the likelihood of each outcome.
- **Random variable**: A variable taking on different values according to chance.
- **Conditional probability**: Probability of an event given another event.

Probability Distribution:
A probability distribution is a function that assigns a probability to each possible outcome of a random experiment. For example, the probability distribution of a coin toss is P(Heads) = 0.5 and P(Tails) = 0.5.

Random Variable:
A random variable is a variable that takes on different values according to chance. For example, let X be the outcome of a die roll. Then X can take on values 1, 2, 3, 4, 5, or 6.

Types of Probability:

- **Marginal Probability**: P(A) is the probability of event A occurring. For example, P(Heads) = 0.5.
- **Conditional Probability**: P(A|B) is the probability of A given B. For example, P(Heads|Coin is fair) = 0.5.

- **Joint Probability**: P(A,B) is the probability of both A and B occurring. For example, P(Heads, Coin is fair) = 0.5.

Bayes' Theorem:
P(A|B) = P(B|A) * P(A) / P(B)
Let's use an example to illustrate this:

- A: Person has disease
- B: Person tests positive
- P(A) = 0.01 (1% of population has disease)
- P(B|A) = 0.99 (99% of people with disease test positive)
- P(B) = 0.05 (5% of population tests positive)

P(A|B) = P(B|A) * P(A) / P(B)
= 0.99 * 0.01 / 0.05
= 0.198

So, given that a person tests positive, there's approximately a 19.8% chance they actually have the disease.

Note:

- **Independence**: P(A,B) = P(A)P(B). For example, coin tosses are independent events.
- **Mutual Exclusivity**: P(A ∩ B) = 0. For example, a coin can't be both Heads and Tails at the same time.

Example: A company has two factories, A and B, producing widgets.

- Factory A produces 60% of the widgets, and 5% of its widgets are defective.
- Factory B produces 40% of the widgets, and 10% of its widgets are defective.

A: Widget is from Factory A

B: Widget is from Factory B

D: Widget is defective

Given:
P(A) = 0.6
P(B) = 0.4
P(D|A) = 0.05
P(D|B) = 0.1
We want to find P(D), the probability that a widget is defective.
Using the law of total probability:
P(D) = P(D|A)P(A) + P(D|B)P(B)
= 0.05 * 0.6 + 0.1 * 0.4
= 0.03 + 0.04
= 0.07
So, the probability that a widget is defective is 7%.
Now, let's use Bayes' Theorem to find P(A|D), the probability that a defective widget is from Factory A:

P(A|D) = P(D|A) * P(A) / P(D)
= 0.05 * 0.6 / 0.07
= 0.4286
So, given that a widget is defective, there's approximately a 42.86% chance it's from Factory A.

Example: A bag contains 5 red balls and 7 green balls.

A: Ball is red
B: Ball is green

Marginal Probability:

P(A) = Number of red balls / Total number of balls = 5/12

P(B) = Number of green balls / Total number of balls = 7/12

Joint Probability:

P(A,B) = P(Ball is both red and green) = 0 (mutually exclusive events)

Conditional Probability:

P(A|B) = P(Ball is red given it's green) = 0 (can't be both)

P(B|A) = P(Ball is green given it's red) = 0 (can't be both)

Bayes' Theorem:

Not applicable here since events are mutually exclusive.

Now, let's calculate the probability of drawing a red ball and then a green ball (without replacement):

$P(A \cap B) = P(A) * P(B|A) = (5/12) * (7/11) = 35/132$

5 Red Balls
7 Green Balls

Information Theory :

Information theory is a branch of mathematics that deals with the quantification, storage, and communication of information. It provides a mathematical framework for understanding the fundamental limits of information processing and transmission.

Concepts:

1. Entropy

- **Definition**: Measures the uncertainty or randomness of a probability distribution.
- **Formula**: $H(X) = - \sum p(x) \log2\ p(x)$
- **Example**: Coin toss: $H(X) = - (0.5 \log2\ 0.5 + 0.5 \log2\ 0.5) = 1$ bit
- **Explanation**: The entropy of a fair coin toss is 1 bit, meaning it takes 1 bit to represent the outcome (Heads or Tails).

2. Joint Entropy

- **Definition**: Measures the uncertainty of two random variables.
- **Formula**: $H(X,Y) = - \sum\sum p(x,y) \log2\ p(x,y)$

- **Example**: Two coin tosses: H(X,Y) = - (0.25 log2 0.25 + 0.25 log2 0.25 + 0.25 log2 0.25 + 0.25 log2 0.25) = 2 bits
- **Explanation**: The joint entropy of two fair coin tosses is 2 bits, meaning it takes 2 bits to represent the outcomes (HH, HT, TH, TT).

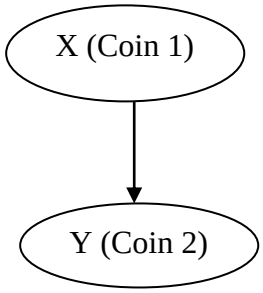
3. Conditional Entropy

- **Definition**: Measures the uncertainty of X given Y.
- **Formula**: H(X|Y) = H(X,Y) - H(Y)
- **Example**: Two coin tosses: H(X|Y) = H(X,Y) - H(Y) = 2 - 1 = 1 bit
- **Explanation**: Given the outcome of the first coin toss, the uncertainty of the second coin toss is 1 bit.

4. Mutual Information

- **Definition**: Measures the dependence between two random variables.
- **Formula**: I(X;Y) = H(X) + H(Y) - H(X,Y)
- **Example**: Two coin tosses: I(X;Y) = H(X) + H(Y) - H(X,Y) = 1 + 1 - 2 = 0 bits
- **Explanation**: The mutual information between two independent coin tosses is 0 bits, meaning they are independent.

Diagram



In this diagram, X and Y are two independent coin tosses. The entropy of X is 1 bit, the entropy of Y is 1 bit, and the joint entropy is 2 bits. The mutual information is 0 bits, indicating independence.

Information theory has many practical applications in various fields. Here are some examples:

1. **Data Compression**: Information theory helps us compress data, like ZIP files, to save storage space and speed up data transfer.
2. **Error-Correcting Codes**: It's used in CDs, DVDs, and wireless communication systems to detect and correct errors, ensuring data integrity.
3. **Communication Networks**: Information theory optimizes network performance, like Wi-Fi signals, to ensure reliable data transmission.

4. **Cryptography**: It's used to create secure encryption methods, like SSL/TLS, to protect online transactions.
5. **Image and Video Compression**: Information theory is used in formats like JPEG and MPEG to compress images and videos.
6. **Bioinformatics**: It's applied to analyze and understand genomic data, like DNA sequencing.
7. **Machine Learning**: Information theory is used in techniques like decision trees and neural networks to improve model performance.

These are just a few examples. Information theory has many more applications in various fields, including finance, economics, and more.

Collocations:

Words or phrases that are often used together in language, forming a common expression.
Example: "Data compression" is a collocation because "data" and "compression" often appear together.
Collocations can be:

- Verb + Noun (e.g., "make a decision")
- Adjective + Noun (e.g., "strong signal")
- Noun + Noun (e.g., "information theory")

Some more examples of collocations:

- **Verb + Noun**:
  - Take a break
  - Make a decision
  - Have a meeting
- **Adjective + Noun**:
  - Strong signal
  - High speed
  - Large dataset
- **Noun + Noun**:
  - Information theory
  - Data compression
  - Error correction
- **Adverb + Adjective**:
  - Highly efficient
  - Very large
  - Extremely complex

Some more examples of collocations in information theory:

- **Channel capacity**
- **Data encryption**
- **Signal processing**
- **Noise reduction**
- **Source coding**

Common Collocations in Information Theory

Some essential collocations used in information theory, along with explanations and examples:

1. **Data Compression**: Reducing the size of data while preserving its content.
   Example: "The ZIP algorithm uses data compression to reduce file sizes."
2. **Error Correction**: Detecting and correcting errors in data transmission or storage.
   Example: "The error correction mechanism ensures data integrity in the communication system."
3. **Information Retrieval**: Retrieving specific data from a large dataset.
   Example: "The search engine uses information retrieval techniques to provide relevant results."
4. **Mutual Information**: Measuring the dependence between two random variables.
   Example: "The mutual information between the two variables indicates a strong correlation."
5. **Data Transmission**: Sending data from one place to another.
   Example: "The data transmission rate is limited by the bandwidth of the channel."
6. **Signal Processing**: Analyzing and modifying signals to extract information.
   Example: "The signal processing algorithm enhances the quality of the audio signal."
7. **Entropy Encoding**: Representing data using entropy-based coding schemes.
   Example: "The Huffman coding algorithm uses entropy encoding to compress the data."

Example Sentences

1. The information theory course covers topics like data compression and error correction.
2. The researcher applied mutual information to analyze the relationship between the variables.
3. The new algorithm improves data transmission rates over wireless networks.
4. Signal processing techniques are used in image and audio compression.
5. Entropy encoding is a fundamental concept in information theory.

N-gram Language Models

An N-gram is a sequence of n items (words, letters, etc.) in a language. N-gram language models predict the next item in a sequence, given the previous n-1 items.

Types of N-grams:

1. **Unigram (1-gram)**: Single word (e.g., "hello")
2. **Bigram (2-gram)**: Two-word sequence (e.g., "hello world")
3. **Trigram (3-gram)**: Three-word sequence (e.g., "hello world example")
4. **N-gram**: General term for n-word sequence

How N-gram Language Models Work:

1. **Training**: Calculate probabilities of word sequences from a large corpus.
2. **Prediction**: Use probabilities to predict the next word in a sequence.

Example:
Corpus: "hello world, hello example, world example"

Bigram probabilities:

- P(world | hello) = 0.5
- P(example | hello) = 0.5
- P(example | world) = 1.0
- Predict next word after "hello": "world" (0.5) or "example" (0.5)

"hello world"
"hello example"
"world example"

Now, let's calculate the probabilities:

P(world | hello):

- "hello" appears 2 times
- "hello" is followed by "world" 1 time
  So, P(world | hello) = 1/2 = 0.5

P(example | hello):

- "hello" appears 2 times
- "hello" is followed by "example" 1 time
  So, P(example | hello) = 1/2 = 0.5

P(example | world):

- "world" appears 1 time
- "world" is followed by "example" 1 time
  So, P(example | world) = 1/1 = 1.0

Now, given the sequence "hello", we want to predict the next word. Based on our calculations:

- P(world | hello) = 0.5
- P(example | hello) = 0.5
- So, the model predicts the next word could be either "world" (with 0.5 probability) or "example" (with 0.5 probability).

Applications:

1. **Language Modeling**: Predicting text, speech, or code.
2. **Text Generation**: Generating text, chatbots, or poetry.
3. **Machine Translation**: Translating text between languages.
4. **Speech Recognition**: Transcribing spoken language.

Estimating Parameters and Smoothing

In N-gram language models, estimating parameters involves calculating probabilities from the training corpus. However, this can lead to issues with unseen events (e.g., a word sequence not present in the training data). Smoothing techniques help address this problem.

Estimating Parameters:

Let's use our previous corpus: "hello world, hello example, world example"

Bigram counts:

- (hello, world): 1
- (hello, example): 1
- (world, example): 1

Unigram counts:

- hello: 2
- world: 2
- example: 2

Estimated bigram probabilities:

- P(world | hello) = count(hello, world) / count(hello) = 1/2 = 0.5
- P(example | hello) = count(hello, example) / count(hello) = 1/2 = 0.5
- P(example | world) = count(world, example) / count(world) = 1/2 = 0.5

Smoothing:

Suppose we want to calculate P(foo | hello). Since "hello foo" is not in the corpus, the maximum likelihood estimate would be 0. Smoothing techniques help assign a non-zero probability to unseen events.

Add-k Smoothing:

- Add a small value k to each count
- P(w | w-1) = (count(w-1, w) + k) / (count(w-1) + k * V)
  Example: k=0.5, V=3
- P(world | hello) = (1 + 0.5) / (2 + 0.5*3) = 1.5 / 3.5 = 0.4286
- P(foo | hello) = (0 + 0.5) / (2 + 0.5*3) = 0.5 / 3.5 = 0.1429

Kneser-Ney Smoothing:

- Discounts the probability of seen events
- Redistributes probability mass to unseen events
- P(w | w-1) = max(count(w-1, w) - δ, 0) / count(w-1) + λ(w-1) * P_continuation(w)

Example: δ=0.5

- P(world | hello) = max(1 - 0.5, 0) / 2 + λ(hello) * P_continuation(world)
- P_continuation(world) = count(world) / total_bigrams = 2 / 3

Laplace Smoothing:
Similar to Add-1, but with a different formulation:

P(foo | hello) = (0 + 1) / (2 + 3) = 1/5

These are just a few examples of smoothing techniques. The goal is to assign reasonable probabilities to unseen events.

Evaluating Language Models

Evaluating language models helps determine their performance and quality. Here are some common metrics:

1. **Perplexity (PPX)**: Measures how well the model predicts a test set.
2. **Cross-Entropy (CE)**: Measures the difference between predicted and actual probabilities.

Test set: "hello world"

Model probabilities:

- P(world | hello) = 0.5
- P(hello) = 0.2 (unigram probability)

Perplexity (PPX):

- PPX = 2^(-1/N * ∑ log2 P(w_i | w_{i-1}))
- N = 2 (number of words: "hello" and "world")
- P(world | hello) = 0.5
- P(hello) = ? (we didn't calculate this, let's assume it's a unigram probability, e.g., P(hello) = 0.2)
  1. log2 P(hello) = log2 0.2 ≈ -2.32
  2. log2 P(world | hello) = log2 0.5 ≈ -1
  3. ∑ log2 P(w_i | w_{i-1}) = -2.32 + (-1) ≈ -3.32
  4. -1/N * ∑ log2 P(w_i | w_{i-1}) = -1/2 * -3.32 ≈ 1.66

5. PPX = 2^(1.66) ≈ 3.16

Cross-Entropy (CE):

- CE = -1/N * ∑ log2 P(w_i | w_{i-1})
- CE ≈ 1.66 (same calculation as above)

Interpretation:

- Perplexity: The model is as good as a random choice among 3.16 possible words.
- Cross-Entropy: The model has an average uncertainty of 1.66 bits per word.

<u>One Mark Question and Answers:</u>

Syntax

_____ is the study of how words are arranged to form sentences.
Answer: Syntax
The _____ of a sentence is a tree-like structure showing its syntactic structure.
Answer: Parse tree

Semantics

_____ is the study of meaning in language.
Answer: Semantics
The _____ of a word is its meaning in a particular context.
Answer: Sense

Pragmatics

_____ is the study of how language is used in context to communicate effectively.
Answer: Pragmatics
The _____ of a sentence is the implied meaning that is not explicitly stated.
Answer: Implicature

Issues - Applications

One major issue in NLP is _____ , or the ability of machines to understand human language.
Answer: Language understanding
NLP has many applications, including _____ , or the translation of text from one language to another.
Answer: Machine translation

The role of machine learning

_____ is a key technique used in NLP to improve the accuracy of language models.
Answer: Machine learning

Probability Basics –Information theory – Collocations -N-gram Language Models

_____ is a type of language model that uses the probability of a word given the previous n-1 words.
Answer: N-gram model

Estimating parameters and smoothing

_____ is a technique used to adjust the probabilities of a language model to avoid zero probabilities.
Answer: Smoothing

Evaluating language models

_____ is a measure of how well a language model predicts a test set.
Answer: Perplexity

****** ALL THE BEST******

VIDYASAGAR COLLEGE OF ARTS AND SCIENCE

UDUMALPET

DEPARTMENT OF DATA SCIENCE

ACADEMIC YEAR : 2025-2026 [EVEN SEMESTER]

CLASS: III B.Sc [DATA SCIENCE]

SUBJECT: NATURAL LANGUAGE PROCESSING

Unit: 2 Morphology and Part of Speech Tagging 14 hours

Linguistic essentials - Lexical syntax- Morphology and Finite State Transducers - Part of speech Tagging - Rule-Based Part of Speech Tagging - Markov Models - Hidden Markov Models – Transformation based Models - Maximum Entropy Models. Conditional Random Fields

.

Morphology and Part of Speech Tagging

In Natural Language Processing (NLP) and linguistics, understanding how words are formed and how they function in sentences is fundamental. **Morphology** and **Part-of-Speech (POS) tagging** are two core concepts that provide this understanding. Morphology deals with the internal structure of words, while POS tagging focuses on identifying the grammatical role of words in a sentence. Together, they form the basis for syntactic and semantic analysis in NLP systems.

Linguistic essentials

Linguistic essentials refer to the fundamental components of language that are necessary for understanding, analyzing, and processing human language. In the context of linguistics and Natural Language Processing (NLP), these essentials describe how language is structured, how meaning is conveyed, and how sentences are formed. A clear understanding of linguistic essentials forms the foundation for higher-level language analysis.

1. Phonetics and Phonology

- **Phonetics** deals with the physical properties of speech sounds.
- **Phonology** studies how sounds function within a particular language.
- These components explain sound patterns, pronunciation, and speech recognition.

Example: The sounds /p/ and /b/ differ in voicing.

2. Morphology

- Morphology studies the internal structure of words and word formation.
- The smallest unit of meaning is called a **morpheme**.
- Includes **inflectional** (tense, number) and **derivational** (word formation) morphology.

Example:

- *unhappiness* = un + happy + ness

3. Syntax

- Syntax deals with sentence structure and the arrangement of words.
- It defines grammatical rules for forming valid sentences.

Example:

- Correct: *She is reading a book.*
- Incorrect: *Reading she book a.*

4. Semantics

- Semantics focuses on the meaning of words, phrases, and sentences.
- It explains relationships such as synonymy, antonymy, and ambiguity.

Example:

- The word *bank* can mean a financial institution or riverbank.

5. Pragmatics

- Pragmatics studies meaning in context.
- It explains how speaker intention and situation affect interpretation.

Example:

- *Can you open the window?* (request, not a question about ability)

6. Discourse Analysis

- Discourse analysis examines language beyond individual sentences.
- It focuses on coherence, cohesion, and context in conversations or texts.

Example:

- Pronoun reference across sentences.

Importance of Linguistic Essentials

- Helps in language understanding and generation
- Essential for NLP tasks such as POS tagging, parsing, machine translation, and speech recognition
- Forms the theoretical backbone of computational linguistics

Linguistic essentials such as phonology, morphology, syntax, semantics, pragmatics, and discourse collectively describe how language works. Understanding these components is crucial for both linguistic analysis and the development of effective NLP systems.

Lexical syntax:

Lexical syntax is a branch of linguistics that studies the relationship between **words (lexicon)** and **sentence structure (syntax)**. It focuses on how lexical items such as nouns, verbs, adjectives, and adverbs influence the grammatical structure of sentences. Lexical syntax explains how words combine according to grammatical rules to form meaningful phrases and sentences.

Lexicon

- The **lexicon** is the mental dictionary of a language.
- It contains information about words, including their meaning, pronunciation, grammatical category, and subcategorization properties.

Example:

- The verb *give* requires three arguments: subject, direct object, and indirect object.

Syntax

- Syntax refers to the set of rules that govern sentence structure.
- It explains word order, phrase structure, and sentence formation.

Example:

- *She reads a book.* (Subject–Verb–Object order)

Relationship between Lexicon and Syntax

Lexical syntax studies how lexical properties determine syntactic behavior.

- Verbs determine sentence patterns (argument structure).
- Nouns determine noun phrase structure.
- Adjectives and adverbs modify nouns and verbs respectively.

Example:

- *sleep* (intransitive verb): *He sleeps.*
- *eat* (transitive verb): *He eats an apple.*

Subcategorization Frames

- Subcategorization refers to the grammatical requirements of a word.
- Verbs specify the number and type of complements they can take.

Examples:

- *put* → requires object and location (*She put the book on the table*)
- *laugh* → no object (*She laughed*)

Lexical Syntax in NLP

In Natural Language Processing, lexical syntax is used in:

- Parsing and syntactic analysis
- Part-of-Speech tagging
- Machine translation
- Information extraction

Lexical information helps NLP systems predict sentence structure accurately.

Importance of Lexical Syntax

- Helps in understanding sentence meaning
- Explains grammatical correctness
- Essential for building language models and parsers

Lexical syntax bridges the gap between vocabulary and grammar. It explains how the properties of words influence sentence structure and meaning. A strong understanding of lexical syntax is essential for linguistic analysis as well as for computational applications such as NLP and machine translation.

Morphology and Finite State Transducers

In linguistics and Natural Language Processing (NLP), **morphology** deals with the internal structure of words, while **Finite State Transducers (FSTs)** are computational models used to implement morphological analysis and generation. FSTs provide an efficient way to model how words are formed by combining roots and affixes according to morphological rules.

Morphology

Morphology is the study of word formation and the relationship between words and morphemes. A **morpheme** is the smallest meaningful unit of a language.

Examples:

- *cats* = cat + s (plural)
- *played* = play + ed (past tense)

Morphology is mainly divided into:

1. **Inflectional morphology** – expresses grammatical features such as tense, number, and person.

2. **Derivational morphology** – forms new words or changes the word class.

Finite State Transducers (FST)

A **Finite State Transducer** is an extension of a Finite State Automaton (FSA). While an FSA accepts or rejects strings, an FST **maps one string to another**.

- FST consists of states, transitions, input symbols, and output symbols.
- Each transition consumes an input symbol and produces an output symbol.
- FSTs are widely used for morphological analysis and word generation.

Role of FST in Morphology

FSTs are used to:

- Analyze surface forms into their morphological components
- Generate surface forms from lexical representations

Example tasks:

- *dogs* → dog + PLURAL
- play + PAST → *played*

Example: Plural Formation Using FST

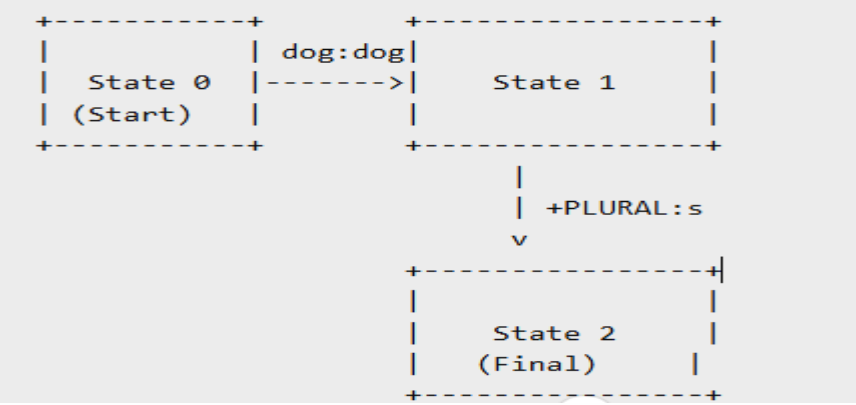
Consider the plural formation of English nouns by adding **-s**.

Lexical Form → Surface Form

dog + PLURAL → dogs

cat + PLURAL → cats

Proper Finite State Transducer (State Diagram)



Explanation of the Diagram:

- **State 0**: Initial state
- **dog:dog** transition copies the root word
- **+PLURAL:s** transition converts the lexical plural marker into surface suffix **-s**
- **State 2**: Final accepting state

Here:

- Input (Lexical): `dog + PLURAL`
- Output (Surface): `dogs`

Morphological Analysis Using FST

For analysis, the FST works in reverse:

dogs → dog + PLURAL

This allows NLP systems to identify the root word and grammatical features.

Advantages of Using FSTs in Morphology

- Efficient and fast
- Bidirectional (analysis and generation)
- Compact representation of morphological rules
- Suitable for languages with rich morphology

Applications in NLP

- Morphological analyzers
- Spell checkers
- Machine translation systems
- Speech recognition

Morphology explains how words are formed using morphemes, while Finite State Transducers provide a formal and computational mechanism to model these processes. By mapping lexical forms to surface forms and vice versa, FSTs play a crucial role in implementing morphological analysis and generation in NLP systems.

Part of speech Tagging

Part-of-Speech (POS) tagging is the process of assigning a grammatical category such as noun, verb, adjective, or adverb to each word in a sentence. It is a fundamental task in linguistics and Natural Language Processing (NLP) because it helps identify how words function in context and supports higher-level language analysis.

Definition of POS Tagging

POS tagging is defined as the task of labeling each word in a sentence with its appropriate part of speech based on both its **definition** and **context**.

Example:

- *book* → noun (*Read a book*)
- *book* → verb (*Book a ticket*)

Common Parts of Speech

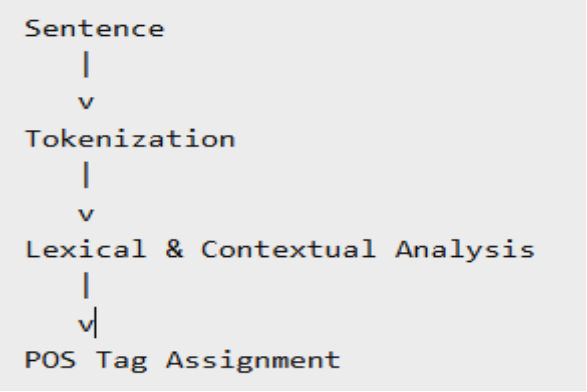
POS Category	Tag	Example
Noun	NN	boy, city
Verb	VB	run, eat
Adjective	JJ	big, happy
Adverb	RB	quickly, very
Pronoun	PRP	he, she
Determiner	DT	a, the
Preposition	IN	in, on
Conjunction	CC	and, but

Example of POS Tagging

Sentence: **"The children are playing in the park."**

Word	POS Tag	Description
The	DT	Determiner
children	NNS	Plural noun
are	VBP	Verb
playing	VBG	Verb (present participle)
in	IN	Preposition
the	DT	Determiner
park	NN	Noun

POS Tagging Process (Diagram)



Approaches to POS Tagging

1. **Rule-Based Approach**
    - Uses handcrafted linguistic rules
    - Accurate but difficult to maintain
2. **Statistical / Probabilistic Approach**
    - Uses probabilities (e.g., Hidden Markov Models)
    - Requires annotated corpora
3. **Machine Learning Approach**
    - Uses algorithms like Decision Trees and Neural Networks
    - More accurate and scalable

Importance of POS Tagging

- Helps in syntactic parsing
- Improves machine translation
- Supports information extraction
- Essential for speech recognition and text analysis

Part-of-Speech tagging identifies the grammatical role of words in a sentence. By combining linguistic rules and statistical methods, POS tagging serves as a foundation for many NLP applications and is a crucial step in understanding natural language.

Rule-Based Part of Speech Tagging

Rule-Based Part-of-Speech (POS) tagging is one of the earliest approaches to POS tagging in linguistics and Natural Language Processing (NLP). In this approach, words in a sentence are assigned POS tags using **handcrafted linguistic rules** based on grammar, word patterns, and context. It relies heavily on expert knowledge of the language.

Rule-Based POS tagging is a method in which POS tags are assigned to words using a set of **manually written rules** that consider:

- Word endings (suffixes/prefixes)
- Word position in a sentence
- Neighboring words and their tags

Components of Rule-Based POS Tagging

1. **Lexicon (Dictionary)**
    - Contains words with their possible POS tags
    - Example: *book* → {NN, VB}
2. **Rule Set**
    - Disambiguation rules to choose the correct tag based on context

Types of Rules

Rule Type	Description	Example
Morphological Rules	Based on word endings	Words ending with *-ly* → Adverb (RB)
Syntactic Rules	Based on sentence structure	DT followed by NN
Contextual Rules	Based on neighboring words	VB after TO

Example Rules

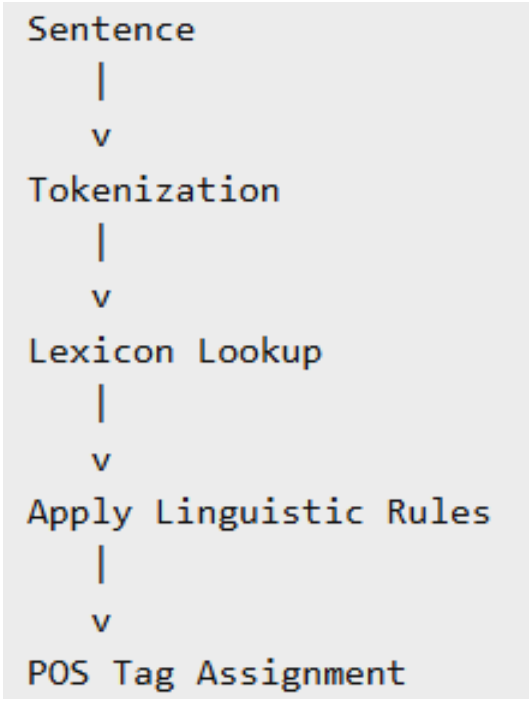
- If a word ends with **-ing**, tag it as **VBG**
- If a word follows a **determiner (DT)**, tag it as **noun (NN)**
- If a word ends with **-ed**, tag it as **past tense verb (VBD)**

Example Sentence Tagging

Sentence: **"The boy is running fast"**

Word	Rule Applied	POS Tag
The	Determiner rule	DT
boy	Follows DT	NN
is	Auxiliary verb rule	VBZ
running	Ends with -ing	VBG
fast	Default adverb rule	RB

Rule-Based POS Tagging Process (Diagram)



Advantages

- Easy to understand and interpret
- No training data required
- Useful for languages with limited corpora

Limitations

- Difficult to write and maintain rules
- Not scalable to large languages
- Lower accuracy for ambiguous words

Rule-Based POS tagging assigns grammatical categories using manually crafted linguistic rules. Although it is less flexible than statistical methods, it forms the foundation of POS tagging and is useful in controlled or low-resource language environments.

Markov Models

A **Markov Model** is a probabilistic model used to represent systems that change states over time. It is based on the **Markov assumption**, which states that the **future state depends only on the present state and not on the sequence of past states**.

Markov Assumption

$$P(S_{t+1} \mid S_t, S_{t-1}, \ldots, S_1) = P(S_{t+1} \mid S_t)$$

This property makes Markov Models simple and efficient for modeling sequential data.

Components of a Markov Model

1. **States (S)** – Possible conditions of the system
2. **Transition Probabilities (A)** – Probability of moving from one state to another
3. **Initial State Distribution (π)** – Probability of starting in each state

Example: Weather Prediction

States

- **Sunny (S)**
- **Rainy (R)**

The weather of tomorrow depends only on today's weather.

Transition Probability Table

Current State	Next: Sunny (S)	Next: Rainy (R)
Sunny (S)	0.7	0.3
Rainy (R)	0.4	0.6

Explanation of Table

- If today is **Sunny**, the probability that tomorrow is:
  - Sunny = 0.7
  - Rainy = 0.3
- If today is **Rainy**, the probability that tomorrow is:
  - Sunny = 0.4
  - Rainy = 0.6

Each row sums to **1**, representing a valid probability distribution.

Initial State Distribution (Example)

State	Probability
Sunny (S)	0.6
Rainy (R)	0.4

Working Example

If today is **Sunny**, the probability that **tomorrow is Rainy** is:

$$P(R \mid S) = 0.3$$

The probability that it is **Sunny for two consecutive days**:

$$P(S \to S) = 0.7 \times 0.7 = 0.49$$

Applications of Markov Models

- Weather forecasting
- Natural Language Processing (POS tagging)
- Speech recognition
- Stock market modeling
- Bioinformatics

A Markov Model provides a simple and effective way to model sequential data using probabilities and the Markov assumption. It forms the foundation for advanced models like **Hidden Markov Models (HMMs)**.

Hidden Markov Models

A **Hidden Markov Model (HMM)** is a statistical model in which the system is assumed to be a **Markov process with hidden states**.
The states are **not directly observable**, but each state produces an **observable output (observation)** with a certain probability.

HMMs are widely used in:

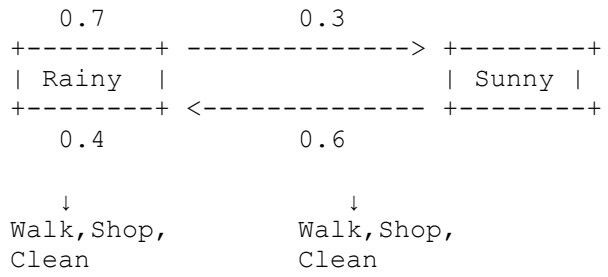
- Speech recognition
- Natural Language Processing (POS tagging)
- Bioinformatics
- Handwriting recognition

Components of an HMM

An HMM is defined by the following elements:

1. **Hidden States (S)**
   Example: {Rainy, Sunny}
2. **Observations (O)**
   Example: {Walk, Shop, Clean}
3. **Initial State Probability (π)**
   Probability of starting in a particular state
4. **State Transition Probability (A)**
   Probability of moving from one hidden state to another
5. **Emission Probability (B)**
   Probability of an observation being generated from a state

HMM State Diagram (Image-style Representation)



Explanation:

- Circles represent **hidden states**
- Arrows represent **state transitions**
- Observations are generated from states but states are hidden

Example Scenario

Weather Prediction based on Activities

- Hidden States:
  S = {Rainy, Sunny}
- Observations:
  O = {Walk, Shop, Clean}

Transition Probability Table (A)

From \ To	Rainy	Sunny
Rainy	0.7	0.3
Sunny	0.4	0.6

Emission Probability Table (B)

State	Walk	Shop	Clean
Rainy	0.1	0.4	0.5
Sunny	0.6	0.3	0.1

Initial Probability Table (π)

State	Probability
Rainy	0.6
Sunny	0.4

Key Characteristics of HMM

- States are **hidden**
- Observations are **visible**
- Follows the **Markov assumption**
- Uses probabilities for prediction and decoding

Applications

- POS Tagging in NLP
- Speech Recognition
- Gene Prediction
- Time Series Analysis

A **Hidden Markov Model** efficiently models systems where the internal states are hidden but can be inferred through observable outputs using probabilistic methods.

Comparison: Markov Model vs Hidden Markov Model (HMM)

Feature	Markov Model (MM)	Hidden Markov Model (HMM)
State Visibility	States are **directly observable**	States are **hidden (not directly observable)**
Observations	Observations are the **states themselves**	Observations are **outputs generated from hidden states**
State Sequence	Known and visible	Unknown and must be inferred
Output Generation	No separate emission process	Uses **emission probabilities**
Probabilities Used	Initial and **transition probabilities only**	Initial, **transition**, and **emission probabilities**
Model Complexity	Simple	More complex
Real-world Modeling	Limited applicability	Models real-world problems effectively
Example	Weather states: Sunny → Rainy	Weather (hidden) → Activities (Walk, Shop)
Algorithms Used	Simple probability calculation	Forward, Backward, Viterbi, Baum-Welch
Applications	Simple sequential data	Speech recognition, POS tagging, bioinformatics

Diagrammatic Difference (Text Representation)

Markov Model

```
scss

Sunny → Rainy → Sunny
(States are visible)
```

Hidden Markov Model

```
scss

Rainy → Sunny → Rainy   (Hidden states)
  ↓       ↓       ↓
 Walk    Shop    Clean  (Observations)
```

- **Markov Model:**
  State = Observation
- **Hidden Markov Model:**
  State ≠ Observation
- HMM is an **extension** of the Markov Model
- HMM is more powerful for **real-world sequence problems**

A **Markov Model** works well when states are observable, whereas a **Hidden Markov Model** is used when states are hidden and only observations are available, making HMM more suitable for complex applications like NLP and speech recognition.

Transformation based Models

Transformation-Based Models are rule-based machine learning models that learn by **iteratively correcting errors** in an initial system using **transformation rules**. They are also known as **Transformation-Based Learning (TBL)** or **Brill's Tagger** in NLP.

These models are widely used in **Part-of-Speech (POS) tagging** and other NLP tasks.

Basic Idea

1. Start with an **initial annotation** (often simple or incorrect)
2. Compare it with **correct (gold standard) data**
3. Learn **transformation rules** that fix the errors
4. Apply rules repeatedly until no improvement is possible

Architecture / Working Diagram (Textual)

```
Raw Text
   ↓
Initial Tagger
   ↓
Error Comparison
   ↓
Learn Transformation Rules
   ↓
Apply Best Rule
   ↓
Final Tagged Output
```

Transformation Rule Format

A typical transformation rule is of the form:

```
Change tag A to tag B
IF condition C is satisfied
```

Example Rule:

```
Change NN to VB
IF the previous word is "to"
```

Example (POS Tagging)

Initial Tagging (Baseline)

Assume every word is tagged as **NN**.

Word	Initial Tag
I	NN
want	NN
to	NN
play	NN

Correct (Gold Standard) Tags

Word	Correct Tag
I	PRP
want	VB
to	TO
play	VB

Learned Transformation Rules

Rule No	Transformation Rule
1	Change NN → PRP if word = "I"
2	Change NN → VB if previous word = "to"
3	Change NN → TO if word = "to"

Final Output After Applying Rules

Word	Final Tag
I	PRP
want	VB
to	TO
play	VB

Algorithm Steps (Training Phase)

1. Assign **initial tags** to all words
2. Compare output with **correct tags**
3. Generate all possible transformation rules
4. Select the rule that gives **maximum error reduction**
5. Apply the rule
6. Repeat until no rule improves accuracy

Advantages

- Produces **human-readable rules**
- High accuracy in POS tagging
- Combines **rule-based and statistical approaches**
- Easy to interpret and debug

Limitations

- Training can be **slow**
- Requires **annotated corpus**
- Performance depends on **initial tagger**

Applications

- Part-of-Speech tagging
- Shallow parsing
- Named Entity Recognition

Transformation-Based Models improve system performance by **learning corrective rules**, making them powerful, interpretable, and effective for NLP tasks such as POS tagging.

Maximum Entropy Models:

Maximum Entropy Models are **probabilistic classification models** used widely in **Natural Language Processing (NLP)** tasks such as **POS tagging, text classification, Named Entity Recognition (NER)**, etc.
They belong to the family of **log-linear models** and make **no independence assumptions** about features.

Principle of Maximum Entropy

The **Maximum Entropy Principle** states:

*Among all probability distributions that satisfy given constraints, choose the one with the **maximum entropy**.*

This ensures:

- The model is **as uniform as possible**
- No additional assumptions are made beyond known evidence
- Avoids bias toward unseen events

Model Definition

For an outcome (label) **y** and context **x**, the conditional probability is:

$$P(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

Where:

- $f_i(x, y)$ → **feature functions** (binary or numeric)
- λ_i → **feature weights**
- $Z(x)$ → **normalization factor**

$$Z(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

Feature Functions

Features capture useful information from input data.

Example (POS Tagging):

Feature Function	Description
$f_1(x, y)$	Word = "run" and tag = Verb
$f_2(x, y)$	Previous tag = Noun
$f_3(x, y)$	Word suffix = "-ing"

- Features are **not required to be independent**
- Multiple overlapping features are allowed

Training of MaxEnt Models

Goal: **Estimate feature weights λi\lambda_iλi**

Training is done by:

- **Maximum Likelihood Estimation**
- Common algorithms:
  - Generalized Iterative Scaling (GIS)
  - Improved Iterative Scaling (IIS)
  - Gradient Descent / L-BFGS

The model adjusts weights so that:

- **Expected feature counts = Observed feature counts**

Advantages of Maximum Entropy Models

- No independence assumption between features
- Can use **rich and overlapping features**
- Works well with sparse and high-dimensional data
- Strong theoretical foundation

Limitations

- Computationally expensive training
- Requires careful feature engineering
- Slower compared to Naïve Bayes
- Needs large training data

Applications

- Part-of-Speech (POS) Tagging
- Named Entity Recognition (NER)
- Text classification
- Word sense disambiguation

Comparison with Naïve Bayes

Aspect	Naïve Bayes	MaxEnt
Feature Independence	Assumed	Not assumed
Model Type	Generative	Discriminative
Flexibility	Low	High
Performance	Fast but limited	Accurate but slower

Maximum Entropy Models provide a **powerful and flexible framework** for NLP tasks by combining probability theory with rich linguistic features, making them highly effective for real-world language processing.

Conditional Random Fields

Conditional Random Fields (CRFs) are **discriminative probabilistic models** used for **sequence labeling tasks** in NLP such as:

- Part-of-Speech (POS) tagging
- Named Entity Recognition (NER)
- Chunking

CRFs model the **conditional probability** of a label sequence given an observation sequence.

Why CRF?

Traditional models like **HMMs** make strong independence assumptions.
CRFs overcome this by:

- Allowing **overlapping and dependent features**
- Avoiding the **label bias problem**
- Directly modeling P(Y|X)

Definition

Given:

- Observation sequence: $X = (x_1, x_2, ..., x_n)$
- Label sequence: $Y = (y_1, y_2, ..., y_n)$

CRF defines:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_t \sum_k \lambda_k f_k(y_{t-1}, y_t, X, t)\right)$$

Where:

- f_k → feature functions
- λ_k → feature weights
- $Z(X)$ → normalization factor

CRF Diagram (Linear Chain CRF)

```
x1        x2        x3        x4
|         |         |         |
y1 ——— y2 ——— y3 ——— y4
```

- xix_ixi: observations (words)
- yiy_iyi: labels (POS/NER tags)
- Edges show **dependencies between adjacent labels**

Feature Functions in CRF

CRF uses two main types of features:

Feature Type	Description
State Feature	Depends on current label and observation
Transition Feature	Depends on previous and current labels

Example Feature Table (NER)

Word	Capitalized	Label
India	Yes	B-LOC
is	No	O
great	No	O

Features can include:

- Current word
- Previous word
- Capitalization
- Prefix / suffix
- Previous label

Advantages of CRF

- No independence assumption between observations
- Supports rich, overlapping features
- Produces globally optimal label sequence
- Better accuracy than HMM and MaxEnt for sequences

Limitations

- Computationally expensive training
- Requires large labeled datasets
- Complex feature engineering

Comparison: HMM vs CRF

Aspect	HMM	CRF
Model Type	Generative	Discriminative
Probability Modeled	P(X,Y)	(P(Y))
Feature Flexibility	Low	High
Label Bias	No	No
Accuracy	Moderate	High

Applications

- Named Entity Recognition (NER)
- POS tagging
- Chunking
- Bioinformatics sequence labeling

Conditional Random Fields are powerful sequence labeling models that combine **probability theory** with **rich linguistic features**, making them highly effective for structured prediction tasks in NLP.

One Mark Question with Answers

Linguistic Essentials

1. _____ is the smallest meaningful unit of language.
   Answer: Morpheme
2. The study of sentence structure is called _____.
   Answer: Syntax
3. The study of meaning is known as _____.
   Answer: Semantics
4. Phonetics deals with the study of _____.
   Answer: Speech sounds
5. Words stored in a language form the _____.
   Answer: Lexicon

Lexical Syntax

1. Lexical syntax deals with the interaction between _____ and syntax.
   Answer: Words

2. The argument structure of a verb specifies its _____.
   Answer: Complements
3. Subcategorization frames describe a word's _____.
   Answer: Syntactic behavior
4. A verb that requires an object is called a _____ verb.
   Answer: Transitive

Morphology and Finite State Transducers (FST)

1. Morphology studies the internal structure of _____.
   Answer: Words
2. Inflectional morphology changes _____ of a word.
   Answer: Grammatical form
3. A Finite State Transducer maps between _____ and surface forms.
   Answer: Lexical
4. FSTs consist of states and _____.
   Answer: Transitions
5. Morphological analyzers are commonly implemented using _____.
   Answer: FSTs

Part of Speech (POS) Tagging

1. POS tagging assigns _____ to words.
   Answer: Grammatical categories
2. Noun, verb, and adjective are examples of _____.
   Answer: POS tags
3. POS tagging helps in resolving _____.
   Answer: Ambiguity
4. The tag **NN** represents a _____.
   Answer: Noun

Rule-Based Part of Speech Tagging

1. Rule-based POS tagging uses hand-crafted _____.
   Answer: Rules
2. These rules are based on _____ knowledge.
   Answer: Linguistic
3. Contextual rules depend on neighboring _____.
   Answer: Words
4. Rule-based taggers do not require _____ data.
   Answer: Training

Markov Models

1. Markov models assume the _____ property.
   Answer: Markov
2. The probability of the next state depends only on the _____ state.
   Answer: Current
3. States in a Markov model are _____.
   Answer: Observable

4. Transition probabilities must sum to _____.
   Answer: One

Hidden Markov Models (HMM)

1. In HMM, states are _____.
   Answer: Hidden
2. Observations are generated from _____.
   Answer: States
3. HMM is widely used for _____ tagging.
   Answer: POS
4. The Viterbi algorithm is used for _____ decoding.
   Answer: Optimal
5. HMM consists of transition and _____ probabilities.
   Answer: Emission

Transformation-Based Models

1. Transformation-based learning was proposed by _____.
   Answer: Brill
2. It starts with an initial _____.
   Answer: Tagging
3. Errors are corrected using _____ rules.
   Answer: Transformation
4. These models combine rule-based and _____ approaches.
   Answer: Statistical

Maximum Entropy Models

1. Maximum Entropy models are _____ classifiers.
   Answer: Probabilistic
2. They choose the model with maximum _____.
   Answer: Entropy
3. MaxEnt models can use many _____.
   Answer: Features
4. They do not assume feature _____.
   Answer: Independence

Conditional Random Fields (CRF)

1. CRF is a _____ discriminative model.
   Answer: Sequence
2. CRF models the conditional probability of _____ given observations.
   Answer: Labels
3. CRFs overcome the _____ bias problem.
   Answer: Label
4. CRF is commonly used for _____ labeling tasks.
   Answer: Sequence

****** ALL THE BEST******