2.3 Text Modelling

2.3.1 Bayesian Networks

The provided text discusses **Bayesian Networks (BNs)** and their application in various domains, including **text mining**.

Bayesian Networks are used to model uncertainty and probabilistic relationships between variables. Key uses include:

1. **Probabilistic Inference**: Predict outcomes based on evidence (e.g., medical diagnosis).
2. **Decision Support**: Evaluate risks and rewards for better decision-making.
3. **Predictive Modeling**: Forecast future events (e.g., weather prediction).
4. **Diagnostics**: Identify causes of system failures (e.g., troubleshooting).
5. **Machine Learning**: Classify data (e.g., spam detection).
6. **NLP**: Text classification and topic detection.
7. **Bioinformatics**: Model gene interactions or disease progression.
8. **Fraud Detection**: Spot suspicious activities.
9. **Robotics**: Decision-making under uncertainty using sensor data.
10. **Causal Modeling**: Analyze cause-and-effect relationships.

Here's a structured overview:

---

1. Definition and Structure

- **Bayesian Networks (BNs)**:
  - Directed Acyclic Graphs (DAGs) where:
    - **Nodes** represent random variables.
    - **Edges** represent conditional dependencies.

---

2. Core Concepts

2.1 Conditional Independence:

- A node is conditionally independent of its **non-descendants**, given its **parents**. This property is called the **local Markov property:**

  $$X_v \perp X_{V \setminus de(v)} \mid X_{pa(v)}, \ \forall v \in V$$

  Where:

  - $de(v)$: Descendants of v.
  - $pa(v)$: Parents of v.

Example (Figure 8.1(a)):

$$x_1 \perp x_3 \mid x_2$$

- X_v: A random variable corresponding to a node v in the graph.
- $X_{V \setminus de(v)}$: The set of random variables corresponding to all nodes in the graph (V) **except** for the descendants ($de(v)$) of node v.
  - $V \setminus de(v)$: The set of nodes excluding the descendants of v.
- $X_{pa(v)}$: The set of random variables corresponding to the **parents** ($pa(v)$) of node v.
- $\perp$: Represents **conditional independence**.
- $|$: "Given" or "Conditioned on".
- $\forall v \in V$: "For all nodes v" in the graph.

2. **Interpretation:**
   - This statement describes the **local Markov property** in a Bayesian network.
   - It reads as:

     > **"The random variable X_v is conditionally independent of all non-descendants of v (denoted $X_{V \setminus de(v)}$), given its parents $X_{pa(v)}$."**

3. **Plain English:**
    - In a Bayesian network, the value of a node depends directly on its **parents.**
    - Once the values of the **parent nodes** ($X_{pa(v)}$) are known, the node (X_v) becomes **independent** of any other node in the graph that is neither a parent nor a descendant of v.
4. **Example:** Consider a simple Bayesian network:
    - $X_1 \to X_2 \to X_3$
    - Here, X_2 has X_1 as its parent and X_3 as its descendant.

    Using the local Markov property:

    - $X_2 \perp X_3 | X_1$: Node X_2 is conditionally independent of its descendant (X_3) given its parent (X_1).

2.2 Factorization Definition:

- The **joint probability distribution (JPD)** of all random variables in a BN can be factored as:

$$p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i \mid pa_i)$$

  Where:

  - pa_i: Set of parent nodes for x_i.

- This differs from the **chain rule:**

$$p(x_1, ..., x_n) = p(x_1)p(x_2|x_1)...p(x_n|x_{n-1}, ..., x_1)$$

- Example distributions for Figure 8.1:
    1. $p(x_1, x_2, x_3) = p(x_1|x_2)p(x_2)p(x_3|x_2)$
    2. $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1, x_3)p(x_3)$
    3. $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2)$

4. Applications in Text Mining

4.1 Spam Filtering:

- A **Naive Bayes classifier** is used to identify spam emails:
    - Words have specific probabilities of appearing in:
        - **Spam** emails (e.g., "free", "credit").

        - **Legitimate** emails (e.g., names of friends).
    - A training set is used to assign spam probabilities to words.
    - If the cumulative spam probability of an email exceeds a threshold, it is marked as spam.

4.2 Information Retrieval:

- BNs are applied to rank and retrieve relevant documents based on probabilistic relationships.

2.3.2 Hidden Markov Model (HMM)

A **Hidden Markov Model (HMM)** is a statistical model used to represent systems that change over time but where the states of the system are hidden (not directly visible). Instead, we observe outputs (or emissions) generated by those hidden states. HMMs are widely used in various fields such as speech recognition, natural language processing, bioinformatics, and time-series analysis.

---

1. Basic Concepts

(a) States:

- The system can exist in a set of **hidden states**, $Y=\{y1,y2,...,yn\}$
- Example: In part-of-speech tagging, the states could be "noun," "verb," or "adjective."

(b) Observations:

- Each state generates an **observation** from a set of possible outputs, $X=\{x1,x2,...,xm\}$
- Example: In part-of-speech tagging, the observations are the words in a sentence.

(c) Hidden:

- The sequence of states that generate the observations is **hidden**, meaning we cannot directly see the states. Instead, we infer them based on the observations.

---

2. Key Components of an HMM

An HMM is fully defined by three sets of parameters, collectively called $\Lambda=\{A,B,\Pi\}$

(a) Transition Probabilities (A):

- These define the probability of transitioning from one state to another.

$$a_{ij} = P(y_{t+1} = q_j \mid y_t = q_i)$$

Example: If the current state is "noun," aij tells us the probability of the next state being "verb."

(b) Emission Probabilities (B):

- These define the probability of an observation being generated from a state.

$$b_i(k) = P(x_t = o_k \mid y_t = q_i)$$

- Example: If the current state is "verb," bi(k) tells us the probability of observing a specific word like "run."

(c) Initial Probabilities (Π):

- These define the probability of starting in a particular state.

$$\pi_i = P(y_0 = q_i)$$

- Example: The probability that the first word in a sentence is a "noun."

---

3. The Two Layers of an HMM

HMMs have two layers:

1. **Hidden States**: Represent the "true" sequence we are trying to uncover (e.g., the parts of speech in a sentence).
2. **Observed Outputs**: Represent the sequence we actually observe (e.g., the words in the sentence).

---

4. Key Properties of HMM

(a) Markov Property:

- The next state depends only on the current state and is independent of previous states.

$$P(y_{t+1} \mid y_t, y_{t-1}, ...) = P(y_{t+1} \mid y_t)$$

(b) Output Independence:

- The observed output depends only on the current state and not on previous observations.

$$P(x_t \mid y_t, x_{t-1}, ...) = P(x_t \mid y_t)$$

---

5. Three Main Problems in HMM

(a) Likelihood Computation:

- Compute the probability of an observed sequence

$$X = \{x_1, ..., x_T\}.$$

- Solved using the **Forward-Backward Algorithm**.

(b) Decoding:

- Find the most likely sequence of hidden states that generated the observations.
- Solved using the **Viterbi Algorithm**.

(c) Learning:

- Estimate the parameters (A,B,ΠA, B, \PiA,B,Π) of the model from observed data.
- Solved using the **Baum-Welch Algorithm** (a special case of the Expectation-Maximization (EM) algorithm).

---

6. Step-by-Step Example

Scenario:

Imagine you are trying to predict the weather (hidden states) based on what your friend tells you they are doing (observations).

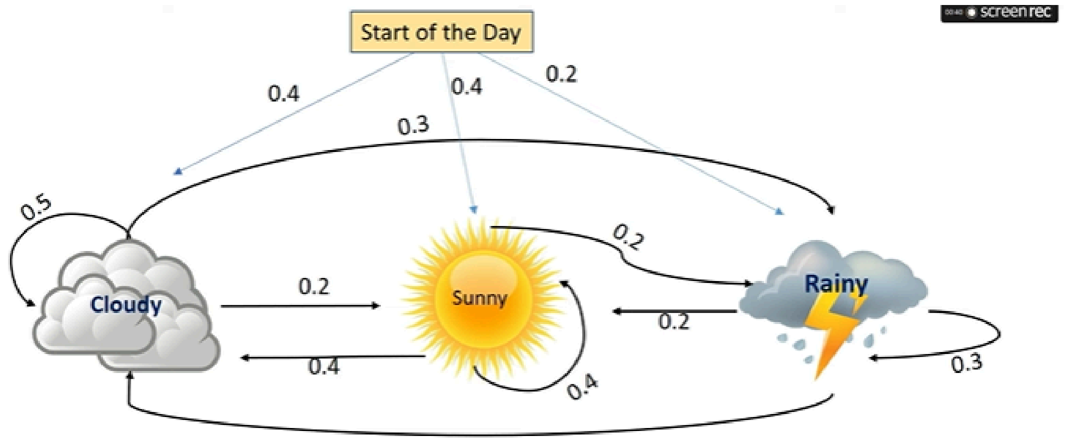
Components:

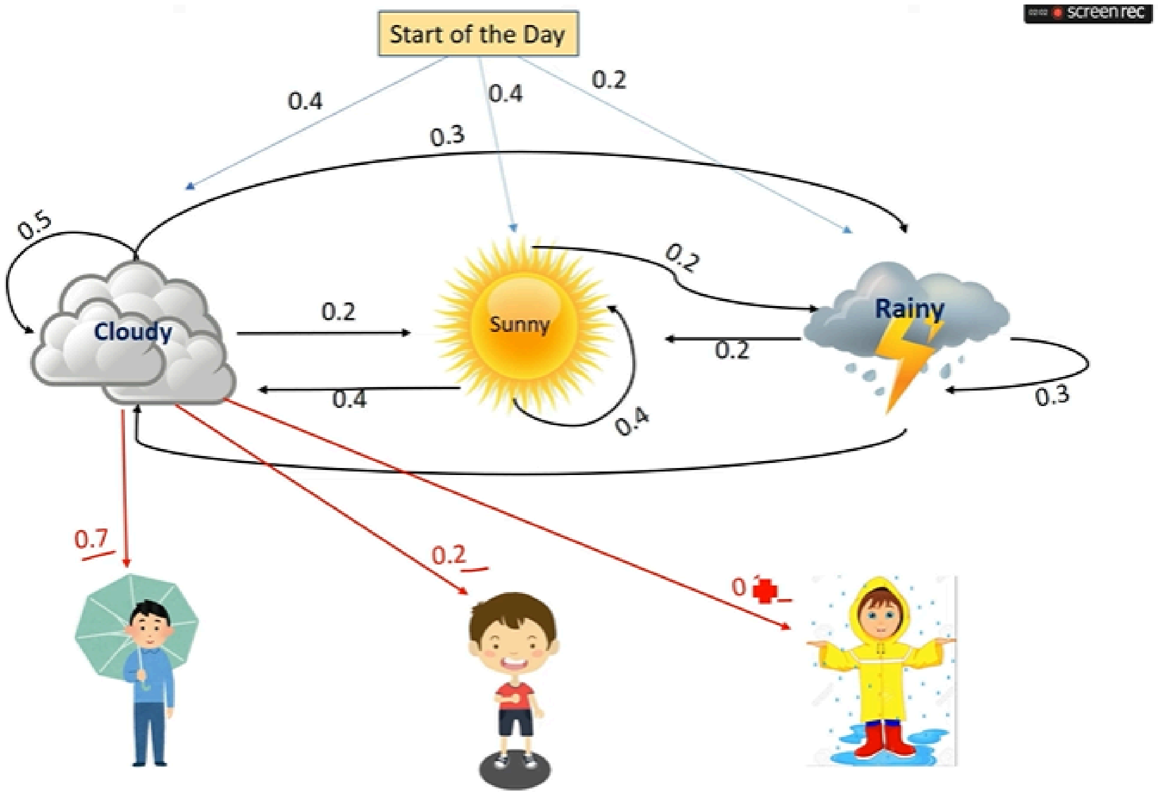
1. **Hidden States:**
   - $Y = \{\text{Sunny}, \text{Rainy}\}$.
2. **Observations:**
   - $X = \{\text{Walk}, \text{Shop}, \text{Clean}\}$.
3. **Transition Probabilities:**
   - If it's sunny today, there's a 70% chance it will be sunny tomorrow and 30% chance it will be rainy.
   - $A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$
4. **Emission Probabilities:**
   - If it's sunny, the friend is likely to walk 80% of the time, shop 10%, and clean 10%.
   - If it's rainy, they clean 50%, shop 40%, and walk 10%.
   - $B = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.4 & 0.5 \end{bmatrix}$
5. **Initial Probabilities:**
   - It starts sunny with 60% probability and rainy with 40%.
   - $\Pi = \{0.6, 0.4\}$.

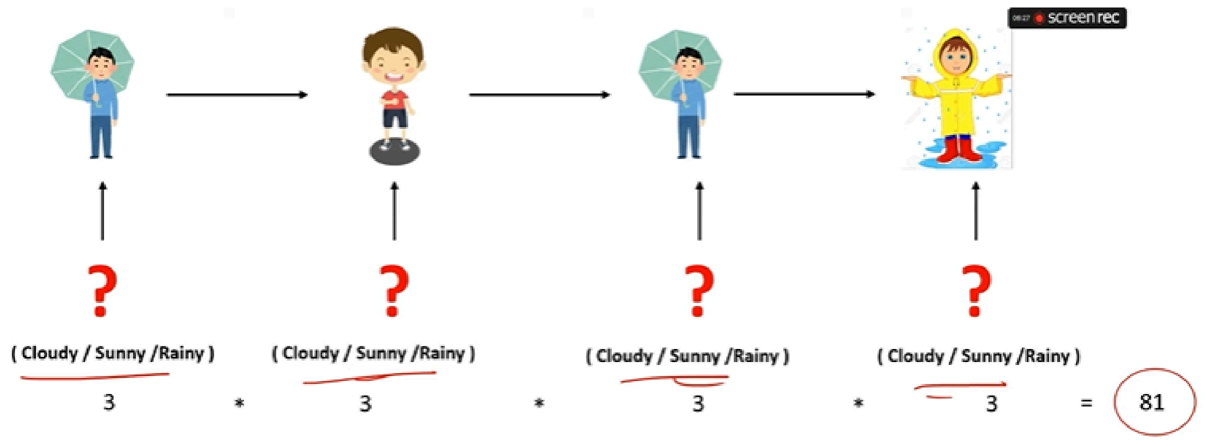
---

7. Practical Applications

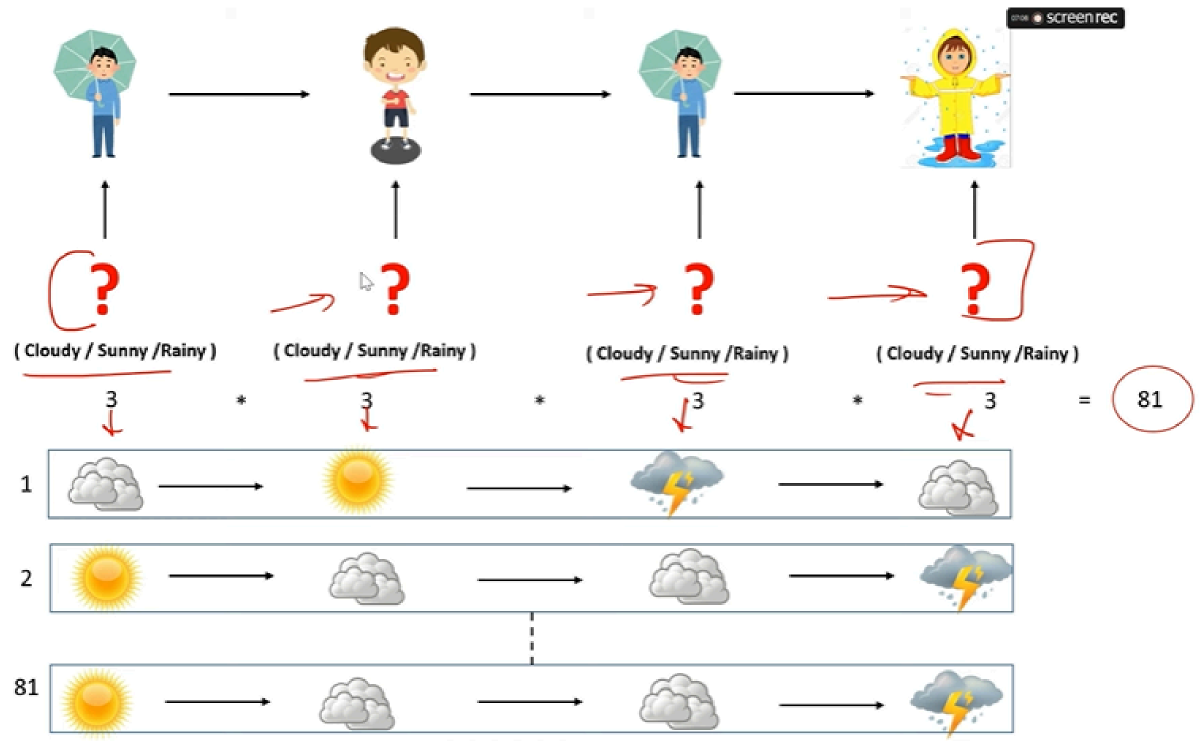
- **Speech Recognition**: HMMs help convert audio signals into text.
- **Part-of-Speech Tagging**: Assign grammatical tags (e.g., noun, verb) to words in a sentence.
- **Named Entity Recognition (NER)**: Identify entities like names, locations, and dates in text.
- **Bioinformatics**: Model gene sequences or protein structures.
- **Time-Series Analysis**: Predict stock prices or weather trends.


Start of the Day
0.4
0.4
0.2
0.3
0.5
Cloudy
0.2
Sunny
0.2
Rainy
0.2
0.4
0.4
0.3

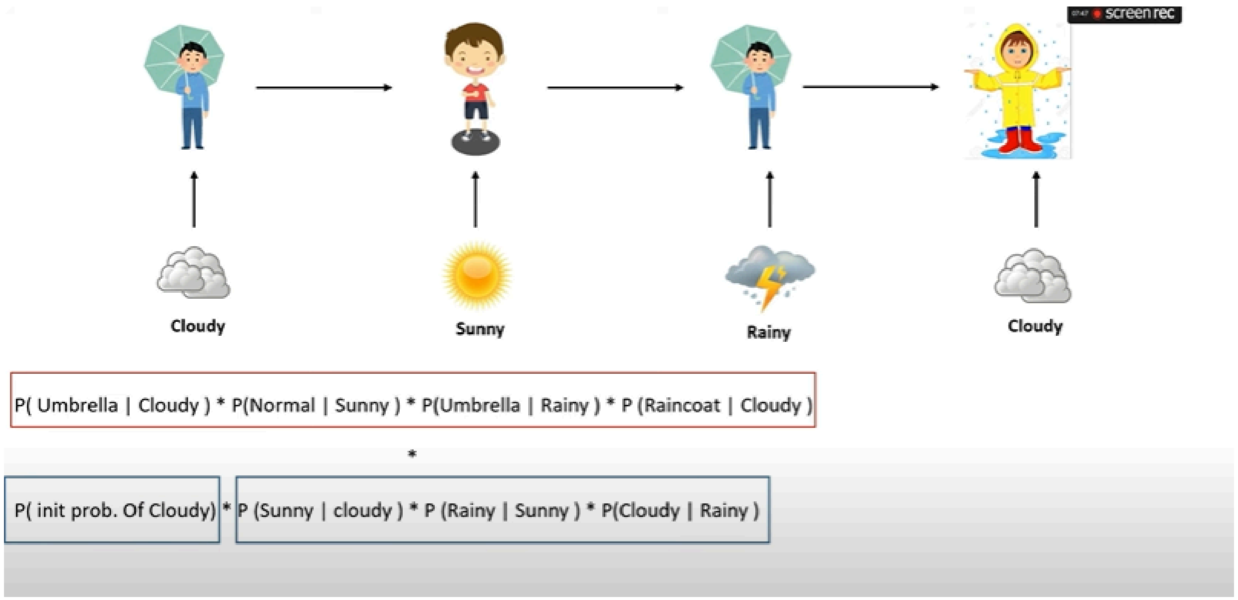

Start of the Day
0.4
0.4
0.2
0.3
0.5
Cloudy
0.2
Sunny
0.2
Rainy
0.2
0.4
0.4
0.3
0.7
0.2
0



screen rec
?
(Cloudy / Sunny /Rainy)
3 * 3 * 3 * 3 = 81



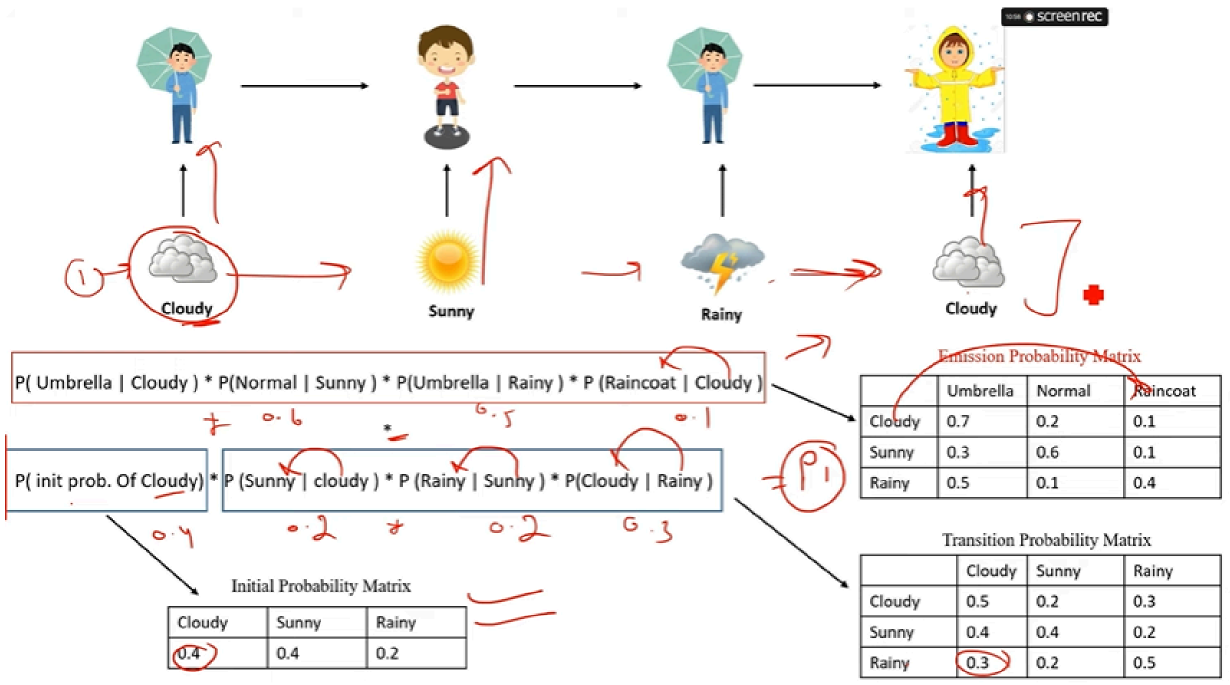
screen rec
?
(Cloudy / Sunny /Rainy)
3 * 3 * 3 * 3 = 81
1
2
81



P(Umbrella | Cloudy) * P(Normal | Sunny) * P(Umbrella | Rainy) * P (Raincoat | Cloudy)

*

P(init prob. Of Cloudy) * P (Sunny | cloudy) * P (Rainy | Sunny) * P(Cloudy | Rainy)



P(Umbrella | Cloudy) * P(Normal | Sunny) * P(Umbrella | Rainy) * P (Raincoat | Cloudy)

* 0.6 0.5 0.1

*

P(init prob. Of Cloudy) * P (Sunny | cloudy) * P (Rainy | Sunny) * P(Cloudy | Rainy) = P_1

0.4 0.2 * 0.2 0.3

Emission Probability Matrix

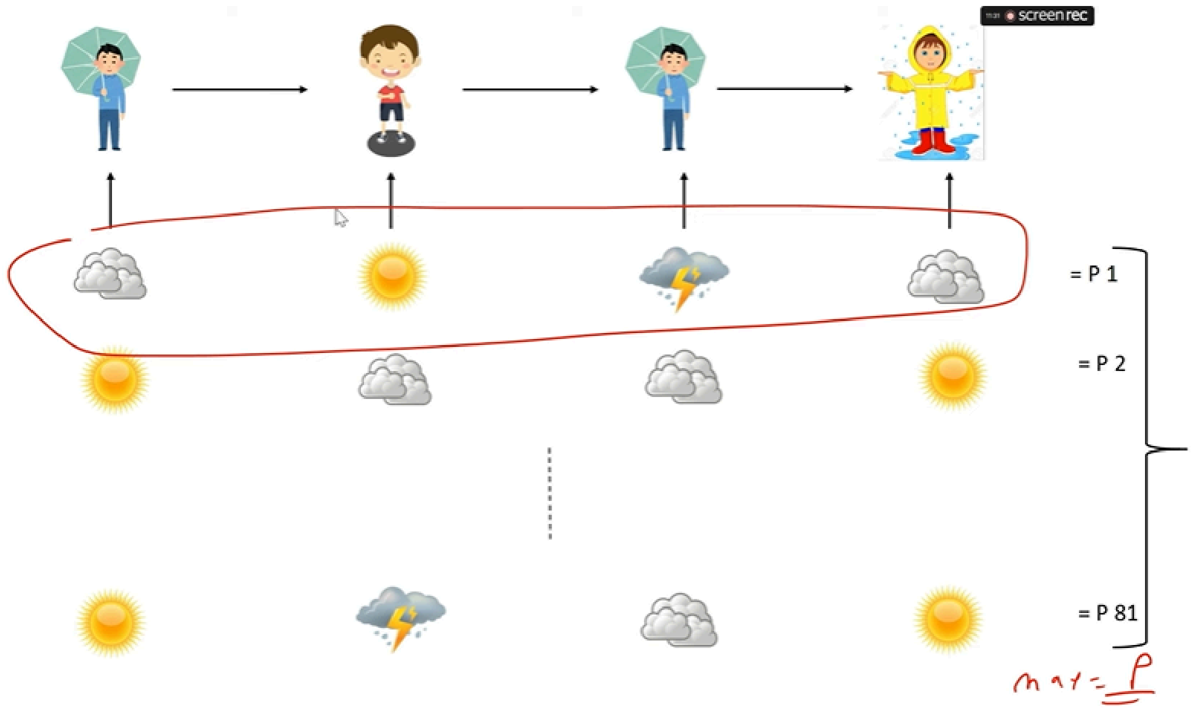
	Umbrella	Normal	Raincoat
Cloudy	0.7	0.2	0.1
Sunny	0.3	0.6	0.1
Rainy	0.5	0.1	0.4

Transition Probability Matrix

	Cloudy	Sunny	Rainy
Cloudy	0.5	0.2	0.3
Sunny	0.4	0.4	0.2
Rainy	0.3	0.2	0.5

Initial Probability Matrix

Cloudy	Sunny	Rainy
0.4	0.4	0.2



Algorithms

Forward Algorithm:

- Computes the probability of an observed sequence P(X∣Λ)
- **Example:**

  Imagine you're observing your friend's activity (walk, shop, clean) and trying to predict if it was sunny or rainy. The Forward Algorithm calculates how likely the sequence of activities (e.g., "walk, clean, shop") is, given the weather patterns in your HMM.

Viterbi Algorithm:

- Finds the most likely sequence of hidden states given observations.
- **Example:**

Given the same observations (e.g., "walk, clean, shop"), the Viterbi Algorithm finds the **most likely sequence of weather states** (e.g., "sunny, rainy, rainy") that produced the activities.

Baum-Welch Algorithm:

- Learns the parameters Λ by maximizing the likelihood of observed sequences.
- **Example:**

  If you're given a dataset of activities (e.g., "walk, clean, shop") and no information about the weather, the Baum-Welch Algorithm can estimate the weather patterns (transition and emission probabilities) that best explain the observations.

Uses of Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) are widely used in various fields for tasks that involve **sequential data** and **probabilistic modeling**. Here's an overview of their key applications:

---

1. Natural Language Processing (NLP)

HMMs are a foundational tool for solving sequence-related tasks in NLP.

Applications:

- **Part-of-Speech (POS) Tagging**: Assign grammatical labels (e.g., noun, verb) to words in a sentence.
  - **Example**: "The cat sleeps" → [The/DT cat/NN sleeps/VB].
- **Named Entity Recognition (NER)**: Identify entities like names, locations, and organizations in text.
  - **Example**: "Barack Obama was born in Hawaii" → [Barack Obama/PER, Hawaii/LOC].
- **Speech Recognition**: Convert spoken language into text by modeling the sequence of phonemes (speech sounds).
  - **Example**: Recognize "hello world" from an audio input.
- **Word Alignment in Translation**: HMMs align words or phrases between languages for tasks like machine translation.

---

2. Speech Processing

HMMs are a cornerstone in processing and analyzing audio data.

Applications:

- **Speech-to-Text Systems**: Recognize spoken words by modeling phoneme sequences.
    - Used in systems like Google Assistant, Siri, and Alexa.
- **Speaker Recognition**: Identify or verify a speaker based on their voice.
- **Language Identification**: Detect the language being spoken from audio input.

---

3. Bioinformatics

HMMs are extensively used to analyze biological sequences.

Applications:

- **Gene Prediction**: Identify coding regions (genes) in DNA sequences.
- **Protein Structure Prediction**: Predict secondary structures of proteins, such as alpha-helices and beta-sheets.
- **Sequence Alignment**: Align DNA, RNA, or protein sequences to find similarities and functional relationships.
- **Protein Family Classification**: Group proteins with similar structures or functions.

---

4. Finance

HMMs model time-series data in finance, which often involves sequences of stock prices or economic indicators.

Applications:

- **Stock Price Prediction**: Model price trends and volatility patterns.
- **Risk Analysis**: Predict and evaluate market risks.
- **Portfolio Management**: Identify hidden states of the market (e.g., bull or bear) to make investment decisions.

---

5. Activity Recognition

HMMs are used in systems that monitor and recognize human actions or behaviors.

Applications:

- **Gesture Recognition**: Identify gestures from motion sensors, such as in gaming or virtual reality (e.g., recognizing a "wave").

- **Video Surveillance**: Detect abnormal activities or patterns in video feeds.
- **Health Monitoring**: Track physical activities for fitness or medical purposes (e.g., identifying walking, running, or resting).

---

6. Robotics

HMMs enable robots to process sensor data and make decisions.

Applications:

- **Path Planning**: Predict the sequence of movements a robot should take.
- **Localization**: Identify the robot's position in a map using sensor readings.
- **Human-Robot Interaction**: Understand gestures, speech, or actions of humans to interact effectively.

---

7. Time-Series Analysis

HMMs model patterns and trends in sequential data over time.

Applications:

- **Weather Prediction**: Model sequences of weather states (e.g., sunny, rainy).
- **Anomaly Detection**: Detect unusual patterns in sensor data, such as in industrial machinery or IoT devices.
- **Medical Diagnosis**: Analyze sequences of symptoms or signals (e.g., ECG data) to detect diseases.

---

8. Games and Entertainment

HMMs enhance user experiences in interactive systems.

Applications:

- **Game AI**: Model player behavior to adapt game difficulty or storyline dynamically.
- **Music Composition**: Generate music or harmonies by learning patterns from existing compositions.

Why HMMs Are Useful

1. **Sequence Modeling**:
    - HMMs handle data that changes over time or follows a sequential order.
2. **Uncertainty Management**:
    - They account for uncertainty by modeling hidden states that can't be directly observed.
3. **Efficient Algorithms**:
    - Algorithms like Forward-Backward, Viterbi, and Baum-Welch make them computationally efficient.
4. **Flexibility**:
    - They can model both discrete and continuous observations.

2.3.3 Markov Random Fields (MRFs)

Overview

- **MRFs**, also called **undirected graphical models**, are described by:
  - **Nodes**: Represent variables or groups of variables.
  - **Links**: Undirected edges representing the relationships between variables.

Key Features:

1. **Conditional Independence:**
   - A set of nodes A and B are conditionally independent given C if C separates A and B in the graph.
   - X_v is conditionally independent of all variables except its neighbors: $X_v \perp X_{V \setminus \{v \cup \mathrm{ne}(v)\}} \mid X_{\mathrm{ne}(v)}$, where $\mathrm{ne}(v)$ denotes the neighbors of node v.
2. **Clique Factorization:**
   - A **clique** is a subset of nodes where every pair is directly connected.
   - The **joint probability distribution** $p(x_1, x_2, \dots, x_n)$ is expressed as:

$$p(x_1, x_2, \dots, x_n) = \frac{1}{Z} \prod_C \psi_C(x_C),$$

   where:

   - $\psi_C(x_C)$: Potential function for clique C.
   - Z: Normalization constant or partition function, defined as $Z = \sum_x \prod_C \psi_C(x_C)$.

$$X_v \perp X_{V \setminus \{v \cup \mathrm{ne}(v)\}} \mid X_{\mathrm{ne}(v)}$$

is a formal way of describing **conditional independence** in a **Markov Random Field (MRF)**. Here's what it means:

Explanation:

1. **Context:**
   - X_v: The variable (or node) v in the graph.
   - X_V: The set of all variables (or nodes) in the graph.
   - $\mathrm{ne}(v)$: The **neighbors** of v, i.e., all nodes that are directly connected to v by an edge.
   - $X_{V \setminus \{v \cup \mathrm{ne}(v)\}}$: All variables except v and its neighbors.

2. **Conditional Independence:**
    - The notation $X_v \perp X_{V \setminus \{v \cup \text{ne}(v)\}} \mid X_{\text{ne}(v)}$ means:
        - The variable X_v is **conditionally independent** of all other variables **outside its Markov blanket** (i.e., the variables not in v or its neighbors) **given the values of its neighbors** $X_{\text{ne}(v)}$.
3. **Interpretation:**
    - In an MRF, a node v only directly depends on its **immediate neighbors** in the graph.
    - Once the states (values) of the neighbors $\text{ne}(v)$ are known, v has no additional dependency on the rest of the graph.

Scenario: Weather Prediction

Imagine a small geographic area where the weather conditions (e.g., rain or no rain) at different locations are interdependent.

Setup:

1. **Nodes**: Each location is represented as a node in the graph.
    - Example: A, B, C, D, E are locations.
2. **Edges**: Each node is connected to its **neighboring locations** (geographically close areas).
    - Example: Locations A and B are neighbors and connected by an edge.

Key Idea:

- If it's raining in A, it's more likely to rain in neighboring locations (B and D).
- If we know the rain status of A's neighbors, A is **conditionally independent** of the rest of the graph.

Conditional Independence:

- The rain at A (R_A) is **independent** of C and E given B and D, its direct neighbors.
- Mathematically:

$$R_A \perp R_{C,E} \mid R_{B,D}$$

Joint Probability Distribution:

The weather (rain status) across all locations (R_A, R_B, R_C, R_D, R_E) can be represented as:

$$p(R) = \frac{1}{Z} \prod_{\text{nodes } i} \psi_i(R_i) \prod_{\text{edges } (i,j)} \psi_{ij}(R_i, R_j),$$

where:

- $\psi_i(R_i)$: Local likelihood of rain at location i.
- $\psi_{ij}(R_i, R_j)$: Relationship (e.g., correlation) between neighboring locations i and j.

Example Values:

1. Suppose:
   - $\psi_{ij}(R_i, R_j) = 1.5$ if $R_i = R_j$ (rain or no rain matches in neighbors).
   - $\psi_{ij}(R_i, R_j) = 0.5$ otherwise (rain mismatch).
2. This encourages neighboring locations to have similar weather.

Given observations like "it's raining at A" and "it's not raining at E", we can use the MRF to infer the likelihood of rain at the remaining locations.

Problem Setup

Nodes and Variables

- A, B, C, D, E: Locations.
- R_A, R_B, R_C, R_D, R_E: Variables representing rain at these locations.
  - $R_i = 1$: Rain at location i.
  - $R_i = 0$: No rain at location i.

Potentials

We define the potentials to model relationships:

1. **Node Potential** (ψ_i):

   Likelihood of rain/no rain at each location.

   - $\psi_A(R_A = 1) = 0.8$, $\psi_A(R_A = 0) = 0.2$ (80% chance of rain at A).
   - Similarly for other nodes.

2. **Edge Potential** (ψ_{ij}):

   Relationship between neighbors (how likely they have the same weather).

   - $\psi_{ij}(R_i = R_j) = 1.5$ (neighbors have the same rain status).
   - $\psi_{ij}(R_i \neq R_j) = 0.5$ (neighbors differ in rain status).

Joint Probability Distribution

The joint probability for rain at all locations $R = (R_A, R_B, R_C, R_D, R_E)$ is:

$$p(R) = \frac{1}{Z} \prod_i \psi_i(R_i) \prod_{(i,j)} \psi_{ij}(R_i, R_j),$$

where Z is the **partition function**, ensuring probabilities sum to 1:

$$Z = \sum_R \prod_i \psi_i(R_i) \prod_{(i,j)} \psi_{ij}(R_i, R_j).$$

Step 2: Compute Joint Probability

$$p(R) = \frac{1}{Z} \cdot (0.8 \cdot 0.7 \cdot 0.6 \cdot 0.9 \cdot 0.4) \cdot (1.5 \cdot 0.5 \cdot 0.5 \cdot 0.5)$$

$$p(R) = \frac{1}{Z} \cdot 0.12096 \cdot 0.1875 = \frac{1}{Z} \cdot 0.02268$$

Step 3: Partition Function Z

To normalize, sum probabilities over all possible assignments of R (i.e., $2^5 = 32$ combinations). For simplicity, we consider only a subset of consistent scenarios here.

Inference

Query: Probability of Rain at B ($R_B = 1$)?

1. Marginalize over all possible configurations of R_A, R_C, R_D, R_E.

$$p(R_B = 1) = \sum_{R_A, R_C, R_D, R_E} p(R).$$

This can be approximated using algorithms like **Loopy Belief Propagation** or **Monte Carlo methods** if the graph is large.

4.3.2 Learning Algorithms

- **Inference Challenges**:
  - Exact inference in MRFs (e.g., summing over all possible assignments) is computationally intractable.
  - Efficient inference methods:
    - **Markov Chain Monte Carlo (MCMC)**.
    - **Loopy Belief Propagation**.

Scenario: Weather and Activities

Imagine you are deciding what activity to do based on the weather and temperature.

We define three random variables:

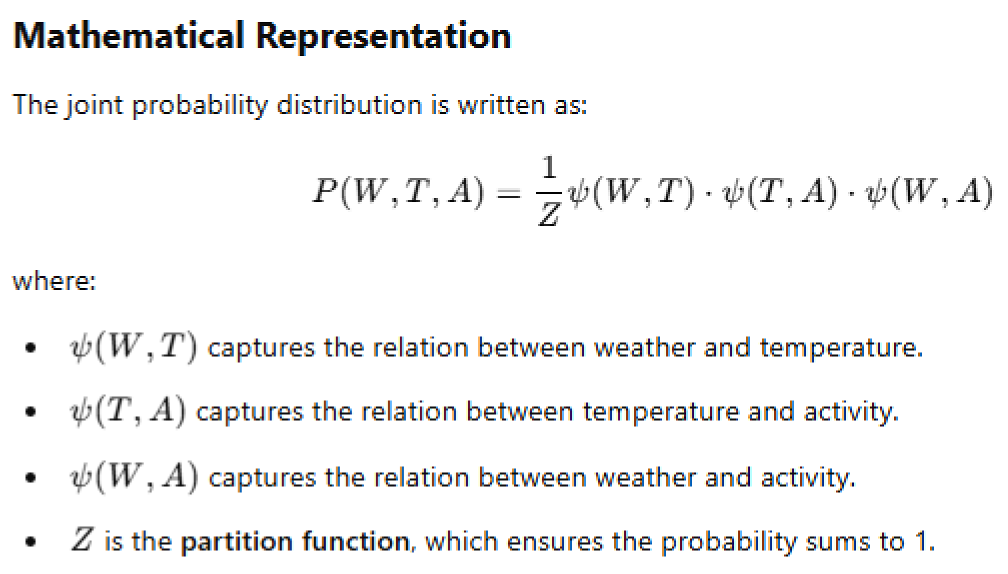
- **W** (Weather: Sunny or Rainy)
- **T** (Temperature: Hot or Cold)
- **A** (Activity: Picnic or Stay Inside)

These variables are **dependent** on each other.

- If the **weather is sunny**, it is more likely that the **temperature is hot**.
- If the **temperature is hot**, you are more likely to go for a **picnic**.
- If the **weather is rainy**, you are more likely to **stay inside**.

Markov Property (Conditional Independence)

- **Activity (A)** depends on **Weather (W) and Temperature (T)**.
- Given **Weather (W), Temperature (T) is conditionally independent of Activity (A)**.
- The **connections** between the variables define a **Markov Random Field**.



Mathematical Representation

The joint probability distribution is written as:

$$P(W,T,A) = \frac{1}{Z}\psi(W,T) \cdot \psi(T,A) \cdot \psi(W,A)$$

where:

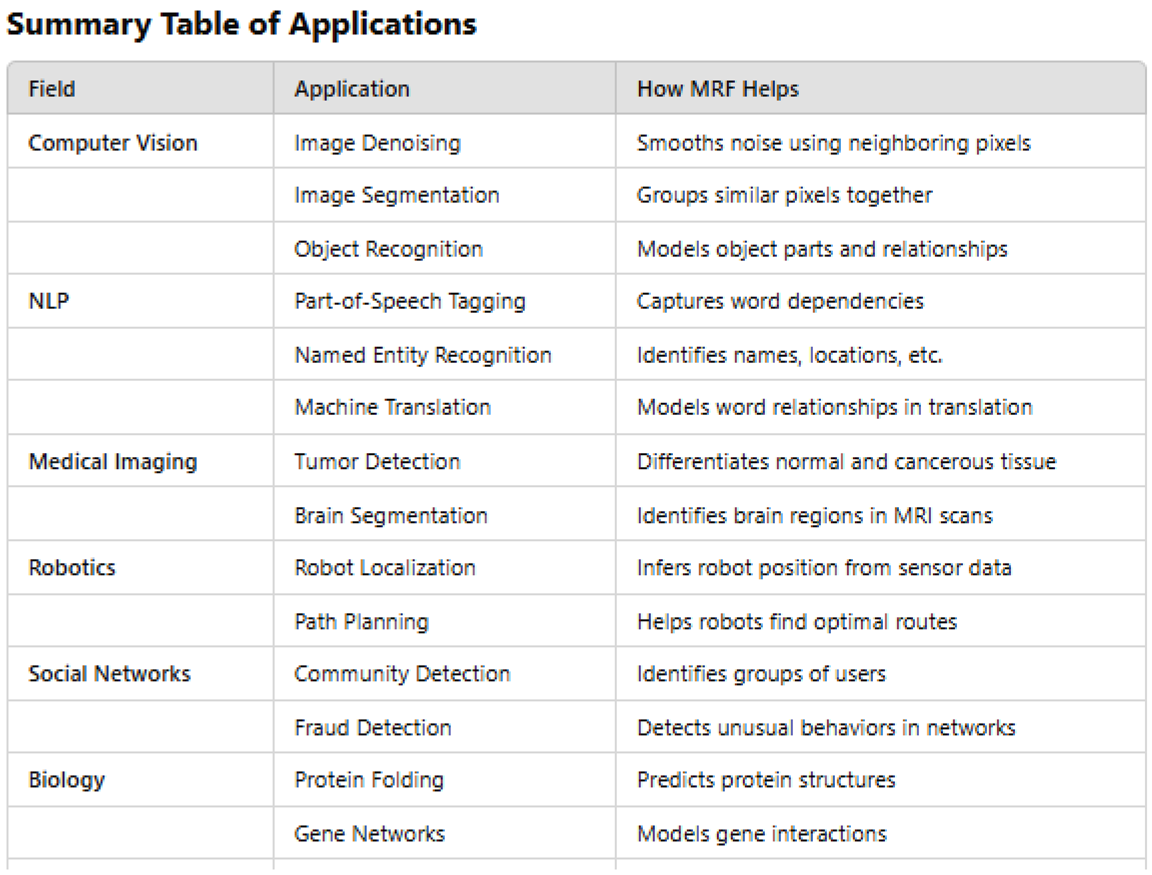
- $\psi(W,T)$ captures the relation between weather and temperature.
- $\psi(T,A)$ captures the relation between temperature and activity.
- $\psi(W,A)$ captures the relation between weather and activity.
- Z is the **partition function**, which ensures the probability sums to 1.

✅ **MRFs are used to model dependencies** between variables.

✅ **Undirected graphs** define relationships between variables.

✅ **Conditional independence** is determined by graph separation.

✅ **Used in real-world applications** like image processing, NLP, and recommendation systems.



Summary Table of Applications

Field	Application	How MRF Helps
Computer Vision	Image Denoising	Smooths noise using neighboring pixels
	Image Segmentation	Groups similar pixels together
	Object Recognition	Models object parts and relationships
NLP	Part-of-Speech Tagging	Captures word dependencies
	Named Entity Recognition	Identifies names, locations, etc.
	Machine Translation	Models word relationships in translation
Medical Imaging	Tumor Detection	Differentiates normal and cancerous tissue
	Brain Segmentation	Identifies brain regions in MRI scans
Robotics	Robot Localization	Infers robot position from sensor data
	Path Planning	Helps robots find optimal routes
Social Networks	Community Detection	Identifies groups of users
	Fraud Detection	Detects unusual behaviors in networks
Biology	Protein Folding	Predicts protein structures
	Gene Networks	Models gene interactions

4.3.3 Applications in Text Mining

- **MRFs in Text Categorization & Information Retrieval**:
    - Model term dependencies in queries and documents.
    - Variants of MRFs (e.g., full independence, sequential dependence) have shown significant improvements in modeling text features and improving **mean average precision** in retrieval tasks.

2.3.4 Conditional Random Fields (CRFs)

Overview

- **CRFs** are a variant of MRFs designed for **sequence labeling** problems.
- Differences from MRFs:
    - CRFs define a **conditional distribution** $p(Y|X)$ instead of a joint distribution $p(Y,X)$
    - No strong independence assumptions on observed variables X.

Applications in Text Mining

- Widely used in tasks like:
    - **POS tagging**.
    - **Shallow parsing**.
    - **Named entity recognition (NER)**.
- CRFs outperform **Hidden Markov Models (HMMs)** in tasks due to their relaxed independence assumptions.
