**📗 Task-03 Report**

**📌 Title: Text Generation with Markov Chains**

**🔹 Introduction**

Markov Chains are mathematical systems that undergo transitions from one state to another on the basis of probabilistic rules.  
In text generation, they are used to predict the **next word or character** based on the previous ones.

**🔹 Objective**

To implement a simple text generation algorithm using Markov Chains that can generate **random but meaningful text** at both **word-level** and **character-level**.

**🔹 Methodology**

1. Input text is tokenized (words or characters).
2. Build **n-grams** and store possible next tokens.
3. Generate text by randomly selecting the next token from the trained model.
4. Provide support for both **word-level** and **character-level** generation.

**🔹 Code Implementation (Excerpt)**

class MarkovChainTextGenerator:

def \_\_init\_\_(self, mode="word", n=2):

self.mode = mode

self.n = n

self.model = defaultdict(list)

def train(self, text):

# tokenization (word or char)

...

def generate(self, length=50, seed=None):

# generate new sequence

...

**🔹 Results**

**Word-level Example:**

artificial intelligence is the future of technology machine learning and deep learning are subsets of ai markov chains are used for text generation

**Character-level Example:**

artificial intellignce is th futue of tecnoogy. marov cains ae usd fr tet genration in natur lague procssing...

**🔹 Applications**

* Chatbots
* Creative writing
* Auto-suggestions
* NLP research

**🔹 Conclusion**

The Markov Chain text generator successfully demonstrates how **statistical models** can be used for **basic text generation**.  
While simple, it forms the **foundation for more advanced NLP models** such as GPT.

**🔹 Author**

👨‍💻 **Sunny Kumar**  
B.Tech CSE 2nd Year | Intern @ PRODIGY InfoTech