import random

import nltk

from nltk.tokenize import word\_tokenize

import PyPDF2

# Download tokenizer data (only first run)

nltk.download("punkt")

class MarkovChainStory:

def \_init\_(self, text, n=2):

"""Initialize with training text and build model."""

self.n = n

self.model = {}

self.build\_model(text)

def build\_model(self, text):

"""Build the Markov chain model from text."""

tokens = word\_tokenize(text.lower())

for i in range(len(tokens) - self.n):

prefix = tuple(tokens[i:i+self.n])

next\_word = tokens[i+self.n]

if prefix not in self.model:

self.model[prefix] = []

self.model[prefix].append(next\_word)

def generate(self, size=50, start\_text=None):

"""Generate text of given size, optionally starting with a prompt."""

if start\_text:

tokens = word\_tokenize(start\_text.lower())

if len(tokens) >= self.n:

prefix = tuple(tokens[-self.n:])

if prefix in self.model:

output = tokens[:] # start with user prompt

else:

output = list(random.choice(list(self.model.keys())))

else:

output = list(random.choice(list(self.model.keys())))

else:

output = list(random.choice(list(self.model.keys())))

# Generate continuation

for \_ in range(size):

prefix = tuple(output[-self.n:])

if prefix in self.model:

next\_word = random.choice(self.model[prefix])

output.append(next\_word)

else:

break

return " ".join(output).capitalize() + "."

# ───────────────────────────────

# Load text from PDF

# ───────────────────────────────

def load\_pdf\_text(path):

pdf\_reader = PyPDF2.PdfReader(path)

text = ""

for page in pdf\_reader.pages:

text += page.extract\_text() + " "

return text

# ───────────────────────────────

# Main program

# ───────────────────────────────

if \_name\_ == "\_main\_":

# Load dataset from PDF

dataset\_text = load\_pdf\_text("fairy\_tales\_dataset.pdf")

# Train model

story\_gen = MarkovChainStory(dataset\_text, n=2)

# Take user prompt

user\_prompt = input("Enter a starting sentence (or leave blank for random): ")

# Generate new story

print("\nGenerated Story:\n")

print(story\_gen.generate(size=120, start\_text=user\_prompt))