**6.  Implement a basic N-gram model for text generation. For example, generate text using a bigram model using python.**

**Program:**

import random

def generate\_bigram\_model(text):

words = text.split()

bigrams = [(words[i], words[i + 1]) for i in range(len(words) - 1)]

model = {}

for word1, word2 in bigrams:

if word1 in model:

model[word1].append(word2)

else:

model[word1] = [word2]

return model

def generate\_text\_bigram(model, num\_words=50):

current\_word = random.choice(list(model.keys()))

text = current\_word

for \_ in range(num\_words - 1):

if current\_word in model:

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

text += " " + next\_word

current\_word = next\_word

else:

break

return text

text = input("Enter your text: ")

model = generate\_bigram\_model(text)

generated\_text = generate\_text\_bigram(model)

print("Generated text:")

print(generated\_text)

**7.  Write a program using the NLTK library to perform part-of-speech tagging on a text.**

**Program:**

import nltk

from nltk.tokenize import word\_tokenize

from nltk import pos\_tag

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

def pos\_tagging(text):

words = word\_tokenize(text)

tagged\_words = pos\_tag(words)

return tagged\_words

text = input("Enter your text: ")

tagged\_text = pos\_tagging(text)

print("Tagged text:")

print(tagged\_text)

**8. Implement a simple stochastic part-of-speech tagging algorithm using a basic probabilistic model to assign POS tags using python.**

**Program:**

import random

class StochasticPOSTagger:

def \_\_init\_\_(self, training\_corpus):

self.word\_tag\_probabilities = self.calculate\_probabilities(training\_corpus)

def calculate\_probabilities(self, training\_corpus):

word\_tag\_counts = {}

tag\_counts = {}

for sentence in training\_corpus:

for word, tag in sentence:

word\_tag\_counts[(word, tag)] = word\_tag\_counts.get((word, tag), 0) + 1

tag\_counts[tag] = tag\_counts.get(tag, 0) + 1

word\_tag\_probabilities = {pair: count / tag\_counts[pair[1]] for pair, count in word\_tag\_counts.items()}

return word\_tag\_probabilities

def tag\_sentence(self, sentence):

tagged\_sentence = []

for word in sentence:

possible\_tags = [tag for (w, tag) in self.word\_tag\_probabilities.keys() if w == word]

if possible\_tags:

selected\_tag = random.choice(possible\_tags)

else:

selected\_tag = 'NOUN'

tagged\_sentence.append((word, selected\_tag))

return tagged\_sentence

training\_corpus = [

[('The', 'DET'), ('cat', 'NOUN'), ('is', 'VERB'), ('on', 'PREP'), ('the', 'DET'), ('mat', 'NOUN')],

[('A', 'DET'), ('dog', 'NOUN'), ('is', 'VERB'), ('running', 'VERB')]

]

tagger = StochasticPOSTagger(training\_corpus)

new\_sentence = ['The', 'dog', 'is', 'running']

tagged\_sentence = tagger.tag\_sentence(new\_sentence)

print(tagged\_sentence)

**9. Implement a rule-based part-of-speech tagging system using regular expressions using python.**

**Program:**

import nltk

from nltk import pos\_tag, word\_tokenize

def perform\_pos\_tagging(text):

words = word\_tokenize(text)

tagged\_words = pos\_tag(words)

return tagged\_words

text = "NLTK is a powerful library for natural language processing."

tagged\_words = perform\_pos\_tagging(text)

print("Original Text:", text)

print("Part-of-Speech Tagging Result:", tagged\_words)

**10. Implement transformation-based tagging using a set of transformation rules, apply a simple rule to tag words using python.**

**Program:**

import nltk

def apply\_rule(word, tag, rules):

for rule in rules:

if word in rule[0]:

if tag in rule[1]:

return rule[1]

return tag

rules = [

(r'\d+', 'CD'),

(r'[a-zA-Z]+(ed|ing|es|)?$', 'VB'),

(r'[a-zA-Z]+(ly|ment)?$', 'RB'),

(r'[a-zA-Z]+(able|ible)?$', 'JJ'),

(r'\b(a|an|the)\b', 'DT'),

(r'[a-zA-Z]+', 'NN')

]

text = "The quick brown fox jumps over the lazy dog"

words = nltk.word\_tokenize(text)

tagged\_words = []

for word in words:

tag = apply\_rule(word, 'NN', rules)

tagged\_words.append((word, tag))

print(tagged\_words)