Program:

import torch

from transformers import BertTokenizer, BertModel

from sklearn.metrics.pairwise import cosine\_similarity

import numpy as np

tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')

model = BertModel.from\_pretrained('bert-base-uncased')

def get\_bert\_embedding(word):

inputs = tokenizer(word, return\_tensors='pt')

with torch.no\_grad():

outputs = model(\*\*inputs)

return outputs.last\_hidden\_state[0][0].numpy()

def calculate\_similarity(word1, word2):

embedding1 = get\_bert\_embedding(word1)

embedding2 = get\_bert\_embedding(word2)

return cosine\_similarity([embedding1], [embedding2])[0][0]

word1 = "king"

word2 = "queen"

similarity = calculate\_similarity(word1, word2)

print(f"BERT Similarity")

print(f"Similarity between '{word1}' and '{word2}': {similarity}")

output:



Program:

import torch

from transformers import RobertaTokenizer, RobertaModel

from sklearn.metrics.pairwise import cosine\_similarity

import numpy as np

tokenizer = RobertaTokenizer.from\_pretrained('roberta-base')

model = RobertaModel.from\_pretrained('roberta-base')

def get\_roberta\_embedding(word):

inputs = tokenizer(word, return\_tensors='pt')

with torch.no\_grad():

outputs = model(\*\*inputs)

return outputs.last\_hidden\_state[0][0].numpy()

def calculate\_similarity(word1, word2):

embedding1 = get\_roberta\_embedding(word1)

embedding2 = get\_roberta\_embedding(word2)

return cosine\_similarity([embedding1], [embedding2])[0][0]

word1 = "king"

word2 = "queen"

similarity = calculate\_similarity(word1, word2)

print(f"Roberta Similarity")

print(f"Similarity between '{word1}' and '{word2}': {similarity}")

output:

