**Name** : Dhanush A

**Roll No**: 717821i111

**Dept** : AD

**Course Code**:21ID31

**Course Name**:Speech and Language Processing

**Assignment** :1

**Word Embeddings and similarity between senetences**

**CODE:**

from transformers import DistilBertTokenizer, DistilBertForQuestionAnswering,DistilBertModel import torch

import numpy as np

from sklearn.manifold import TSNE

import matplotlib.pyplot as plt

from sklearn.metrics.pairwise import cosine\_similarity

**# Load pre-trained model and tokenizer**

tokenizer1 = DistilBertTokenizer.from\_pretrained('distilbert-base-cased')

model1 = DistilBertModel.from\_pretrained('distilbert-base-cased')

def encode\_sentence(sentence):

**# Tokenize input**

inputs = tokenizer1(sentence, return\_tensors='pt')

**# Get model output**

outputs = model1(\*\*inputs)

**# Extract the embeddings from 'last\_hidden\_state'**

embeddings = outputs.last\_hidden\_state

return embeddings

def encode\_sentence\_sim(sentence, max\_length=512):

**# Tokenize input**

inputs = tokenizer1(sentence, return\_tensors='pt', truncation=True, max\_length=max\_length) **# Get model output**

outputs = model1(\*\*inputs)

**# Extract the embeddings from 'last\_hidden\_state'**

embeddings = outputs.last\_hidden\_state

**# Reduce embeddings to a fixed size (e.g., by mean pooling)**

embeddings = torch.mean(embeddings, dim=1)

return embeddings

def visualize\_word\_embeddings(embeddings, tokens):

**# Use t-SNE to reduce dimensionality for visualization**

tsne\_model = TSNE(n\_components=2, random\_state=42, perplexity=min(5, len(tokens)-1)) word\_vectors\_2D = tsne\_model.fit\_transform(embeddings)

**# Plotting the words in 2D**

plt.figure(figsize=(8, 6))

for i, token in enumerate(tokens):

plt.scatter(word\_vectors\_2D[i, 0], word\_vectors\_2D[i, 1], marker='o', color='b') plt.text(word\_vectors\_2D[i, 0] + 0.02, word\_vectors\_2D[i, 1] + 0.02, token, fontsize=9)

plt.show()

def calculate\_cosine\_similarity(embeddings1, embeddings2):

**# Detach tensors before converting to numpy arrays**

embeddings1 = embeddings1.detach().numpy()

embeddings2 = embeddings2.detach().numpy()

**# Reshape embeddings if needed**

embeddings1 = embeddings1.reshape(1, -1)

embeddings2 = embeddings2.reshape(1, -1)

**# Calculate cosine similarity**

similarity\_matrix = cosine\_similarity(embeddings1, embeddings2)

**# Extract the similarity score**

similarity\_score = similarity\_matrix[0, 0]

return similarity\_score

**# Example usage**

context =input()

question =input()

**# Encode the sentence to get the embeddings**

embeddings = encode\_sentence(question)

**# Extract word vectors directly from the model**

word\_vectors = embeddings[0].detach().numpy()

# Example usage

context\_embeddings = encode\_sentence\_sim(context)

question\_embeddings = encode\_sentence\_sim(question)

similarity\_score = calculate\_cosine\_similarity(context\_embeddings, question\_embeddings) print(f"Cosine Similarity between context and question: {similarity\_score}")

**# Tokenize input for visualization**

tokens = tokenizer.tokenize(tokenizer.decode(tokenizer.encode(question)))

**# Visualize word embeddings**

print("visualization of word embedding")

visualize\_word\_embeddings(word\_vectors, tokens)

**Requirements.txt**

streamlit

transformers

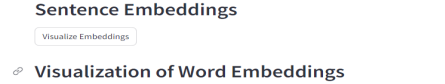
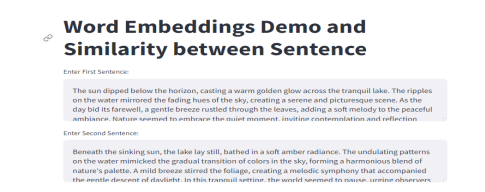
torch

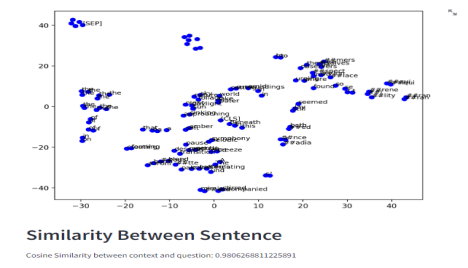
numpy

scikit-learn

matplotlib

**OUTPUT:**

****

****

Github link : https://github.com/DhanushAshok04/wordembedding

Streamlit link: https://wordembedding-cosinesimilarity.streamlit.app/

Colab link : https://colab.research.google.com/drive/1YYSlxNRnNqJfKajUiOaPxTyoAoz4RoEn#scrollTo=Iz\_J2-wCQ-af