Experiment 2: Use dimensionality reduction (e.g., PCA or t-SNE) to visualize word embedding for,

1. Select 10 words from a specific domain (e.g., sports, technology) and visualize their embedding.
2. Analyse clusters and relationships.
3. Generate contextually rich outputs using embedding.
4. Write a program to generate 5 semantically similar words for a given input.

Experiment: Visualizing Word Embedding's using PCA and t-SNE

I. Introduction

Word embedding's are a fundamental concept in **Natural Language Processing (NLP)**, where words are represented as high-dimensional numerical vectors. **Word2Vec**, developed by Google, is a model that learns word representations from a large text corpus. In this experiment, we will use a **pre-trained Word2Vec model** to analyze the relationships between words, visualize them in two-dimensional space, and find semantically similar words.

To handle large word embedding models efficiently, we will use **Google Drive** to store and retrieve the Word2Vec model, preventing redundant downloads.

II. Objectives

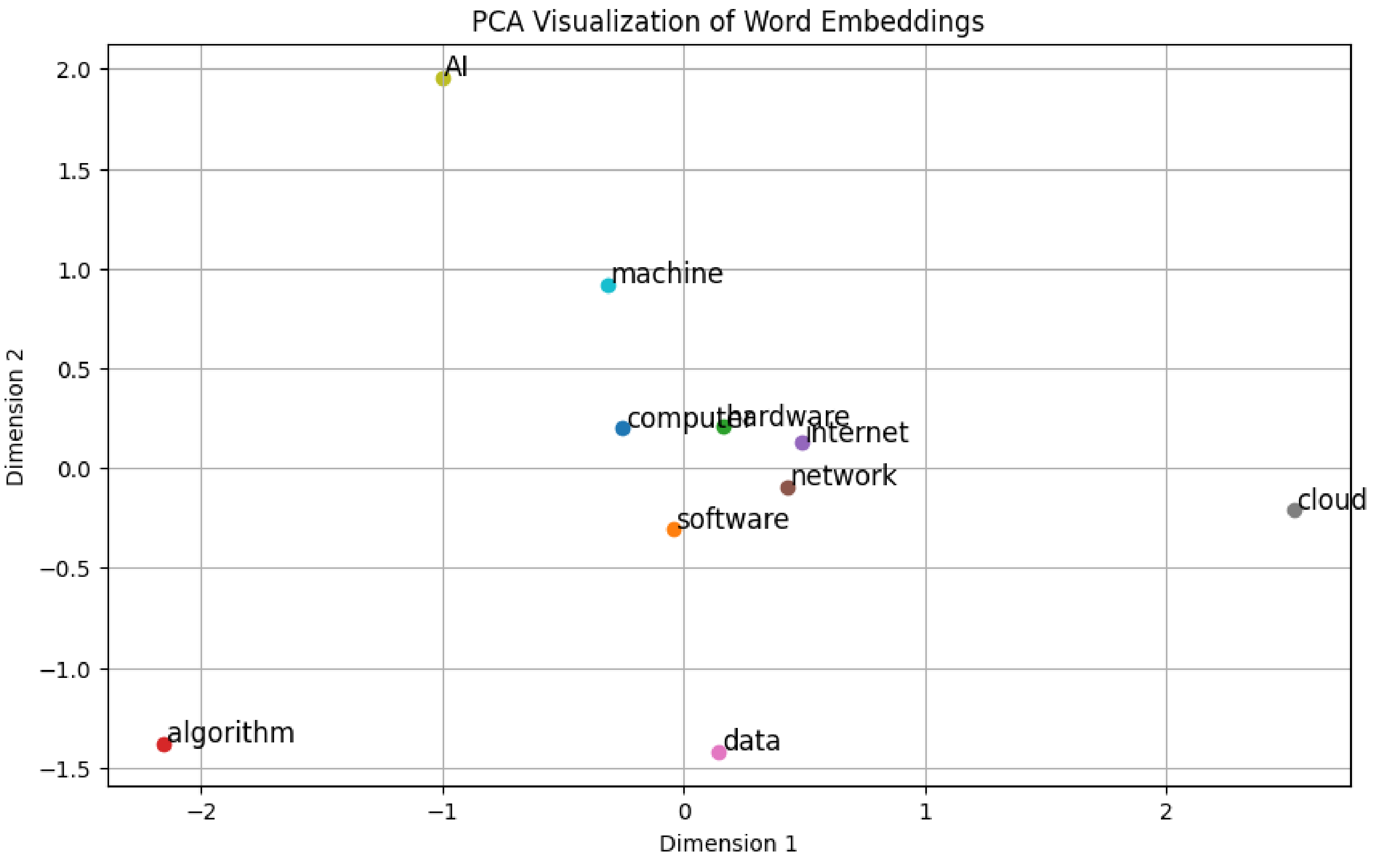
1. **Understand Word Embeddings** – Learn how words are represented as numerical vectors.
2. **Use Word2Vec Model** – Load a **pre-trained Word2Vec model** stored in **Google Drive** or download it if necessary.
3. **Visualize Word Relationships** – Reduce the dimensionality of word vectors using **PCA** and **t-SNE** to create a 2D visualization.
4. **Analyze Word Similarity** – Identify **the most similar words** to a given input word based on its vector representation.
5. **Efficient Model Storage** – Store and retrieve the Word2Vec model from **Google Drive** to optimize performance in Google Colab.

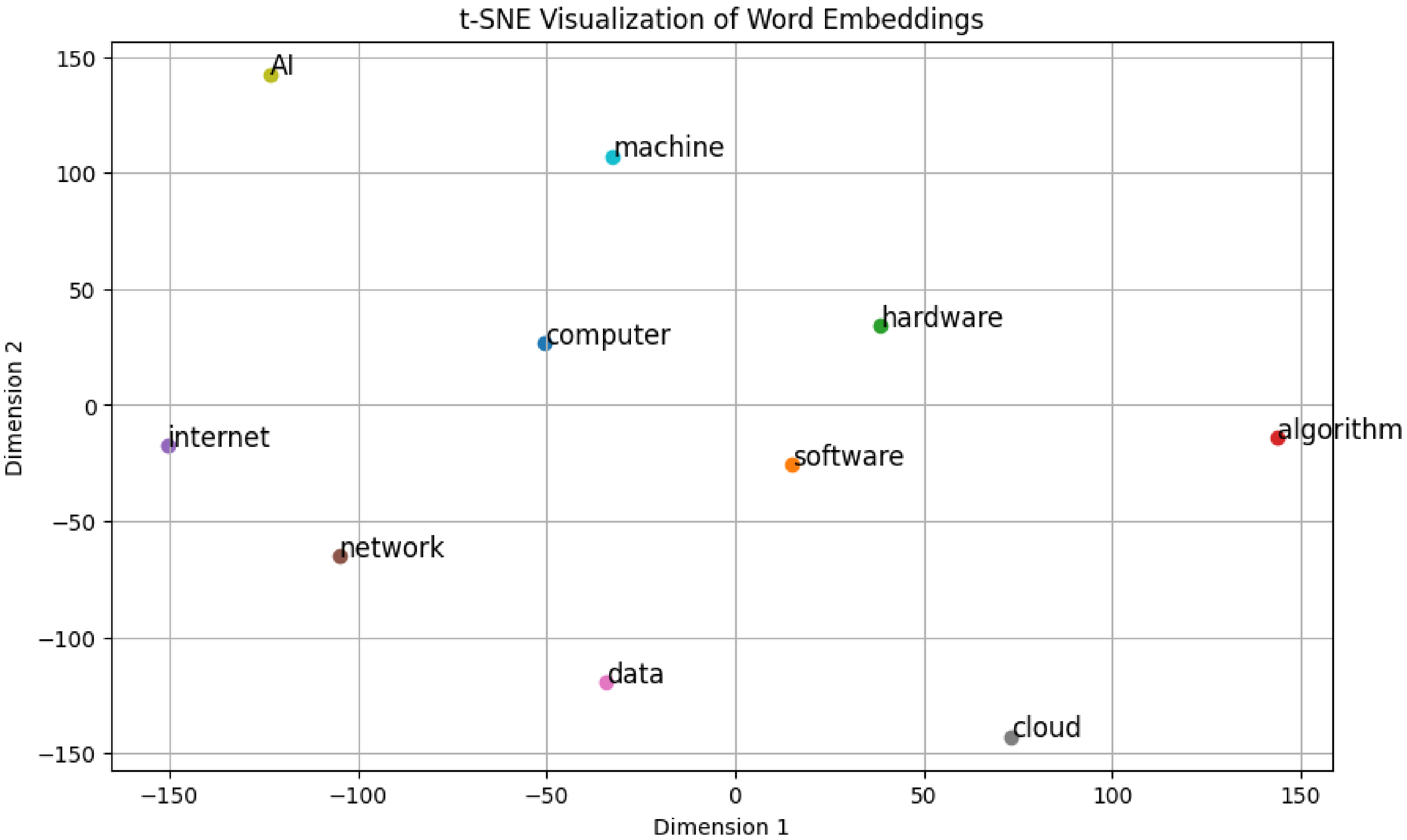
```
import gensim
import gensim.downloader as api
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import os

# Define model path in Google Drive
model_path = "/content/drive/My Drive/word2vec-google-news-300.model"

# Step 1: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Step 2: Check if model already exists
if os.path.exists(model_path):
    print(" Model found! Loading the saved model...")
    word2vec_model = gensim.models.KeyedVectors.load(model_path, mmap='r')
else:
    print("Model not found. Downloading now...")
    word2vec_model = api.load("word2vec-google-news-300")
    # Save the downloaded model to Google Drive
    print("Saving model to Google Drive...")
    word2vec_model.save(model_path)
    print(" Model saved successfully!")

def get_word_vectors(model, words):
   return np.array([model[word] for word in words if word in model])

def reduce_dimensions(vectors, method='pca'):
    if method == 'pca':
        reducer = PCA(n_components=2)
    elif method == 'tsne':
        reducer = TSNE(n_components=2, random_state=42, perplexity=5)
    else:
        raise ValueError("Method should be 'pca' or 'tsne'")
    return reducer.fit_transform(vectors)

def plot_embeddings(words, reduced_vectors, title):
    plt.figure(figsize=(10, 6))
    for word, coord in zip(words, reduced_vectors):
        plt.scatter(coord[0], coord[1], marker='o')
        plt.text(coord[0] + 0.01, coord[1] + 0.01, word, fontsize=12)
    plt.title(title)
    plt.xlabel("Dimension 1")
    plt.ylabel("Dimension 2")
    plt.grid()
    plt.show()

def find_similar_words(model, word, top_n=5):

    if word in model:
        similar_words = model.most_similar(word, topn=top_n)
        return [w[0] for w in similar_words]
    else:
        return ["Word not in vocabulary"]

# Define 10 words from the technology domain
tech_words = ["computer", "software", "hardware", "algorithm", "internet",
              "network", "data", "cloud", "AI", "machine"]
# Get word embeddings
print("Fetching word embeddings...")
word_vectors = get_word_vectors(word2vec_model, tech_words)

# Reduce dimensions using PCA
```

```
print("Applying PCA...")
reduced_vectors_pca = reduce_dimensions(word_vectors, method='pca')
plot_embeddings(tech_words, reduced_vectors_pca, title="PCA Visualization of Word Embeddings")

# Reduce dimensions using t-SNE
print("Applying t-SNE...")
reduced_vectors_tsne = reduce_dimensions(word_vectors, method='tsne')
plot_embeddings(tech_words, reduced_vectors_tsne, title="t-SNE Visualization of Word Embeddings")

# Find similar words
input_word = "computer"
print(f"Finding words similar to '{input_word}'...")
similar = find_similar_words(word2vec_model, input_word)
print("Top similar words:", similar)
```

III. Output

```
Mounted at /content/drive
Model found! Loading the saved model...
Fetching word embeddings...
Applying PCA...
```


PCA Visualization of Word Embeddings
Dimension 1
Dimension 2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
AI
machine
computer
hardware
internet
network
software
cloud
algorithm
data


t-SNE Visualization of Word Embeddings
150
100
50
0
−50
−100
−150
Dimension 2
−150
−100
−50
0
50
100
150
Dimension 1
AI
machine
computer
hardware
internet
software
algorithm
network
data
cloud