```
# Install gensim library
!pip install gensim

# Loading pre-trained word embeddings
import gensim.downloader as api

# Numerical operations
import numpy as np

# Optional data handling
import pandas as pd

# Visualization library
import matplotlib.pyplot as plt

# t-SNE for dimensionality reduction of high-dimensional data
from sklearn.manifold import TSNE
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.4.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (2.0.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.16.3)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim) (2.1.1)
```

```
# Load pre-trained GloVe model with 100 dimensions from gensim's API
model = api.load("glove-wiki-gigaword-100")

# Print the total number of words in the loaded vocabulary
print("Vocabulary Size:", len(model))

# Display the vector representation for the word 'computer' as an example
print("\nVector for word 'computer':")
print(model['computer'])
```

```
Vocabulary Size: 400000

Vector for word 'computer':
[-1.6298e-01  3.0141e-01  5.7978e-01  6.6548e-02  4.5835e-01 -1.5329e-01
  4.3258e-01 -8.9215e-01  5.7747e-01  3.6375e-01  5.6524e-01 -5.6281e-01
  3.5659e-01 -3.6096e-01 -9.9662e-02  5.2753e-01  3.8839e-01  9.6185e-01
  1.8841e-01  3.0741e-01 -8.7842e-01 -3.2442e-01  1.1202e+00  7.5126e-02
  4.2661e-01 -6.0651e-01 -1.3893e-01  4.7862e-02 -4.5158e-01  9.3723e-02
  1.7463e-01  1.0962e+00 -1.0044e+00  6.3889e-02  3.8002e-01  2.1109e-01
 -6.6247e-01 -4.0736e-01  8.9442e-01 -6.0974e-01 -1.8577e-01 -1.9913e-01
 -6.9226e-01 -3.1806e-01 -7.8565e-01  2.3831e-01  1.2992e-01  8.7721e-02
  4.3205e-01 -2.2662e-01  3.1549e-01 -3.1748e-01 -2.4632e-03  1.6615e-01
  4.2358e-01 -1.8087e+00 -3.6699e-01  2.3949e-01  2.5458e+00  3.6111e-01
  3.9486e-02  4.8607e-01 -3.6974e-01  5.7282e-02 -4.9317e-01  2.2765e-01
  7.9966e-01  2.1428e-01  6.9811e-01  1.1262e+00 -1.3526e-01  7.1972e-01
 -9.9605e-04 -2.6842e-01 -8.3038e-01  2.1780e-01  3.4355e-01  3.7731e-01
 -4.0251e-01  3.3124e-01  1.2576e+00 -2.7196e-01 -8.6093e-01  9.0053e-02
 -2.4876e+00  4.5200e-01  6.6945e-01 -5.4648e-01 -1.0324e-01 -1.6979e-01
  5.9437e-01  1.1280e+00  7.5755e-01 -5.9160e-02  1.5152e-01 -2.8388e-01
  4.9452e-01 -9.1703e-01  9.1289e-01 -3.0927e-01]
```

```
# Define a list of words categorized by type for analysis
words = [
    # Animals
    "cat", "dog", "lion", "tiger", "elephant", "horse", "wolf", "monkey",

    # Fruits
    "apple", "banana", "mango", "grape", "orange", "pineapple", "pear", "peach",

    # Countries
    "india", "china", "france", "germany", "italy", "japan", "brazil", "canada",

    # Technology
    "computer", "laptop", "keyboard", "mouse", "internet", "software", "hardware", "mobile",

    # Vehicles
    "car", "bus", "truck", "bicycle", "motorcycle", "train", "airplane", "ship"
]

# Extract the word vectors for the defined list of words using the pre-trained model
# and convert them into a NumPy array for further processing.
word_vectors = np.array([model[word] for word in words])
```

```
# Initialize t-SNE with 2 components for 2D visualization,
# a fixed random state for reproducibility, and perplexity to balance local and global aspects.
tsne = TSNE(n_components=2, random_state=42, perplexity=10)
```

```
# Reduce the dimensionality of the word vectors using the initialized t-SNE model
reduced_vectors = tsne.fit_transform(word_vectors)

# Extract the x and y coordinates from the reduced 2D vectors
x_coords = reduced_vectors[:, 0]
y_coords = reduced_vectors[:, 1]
```

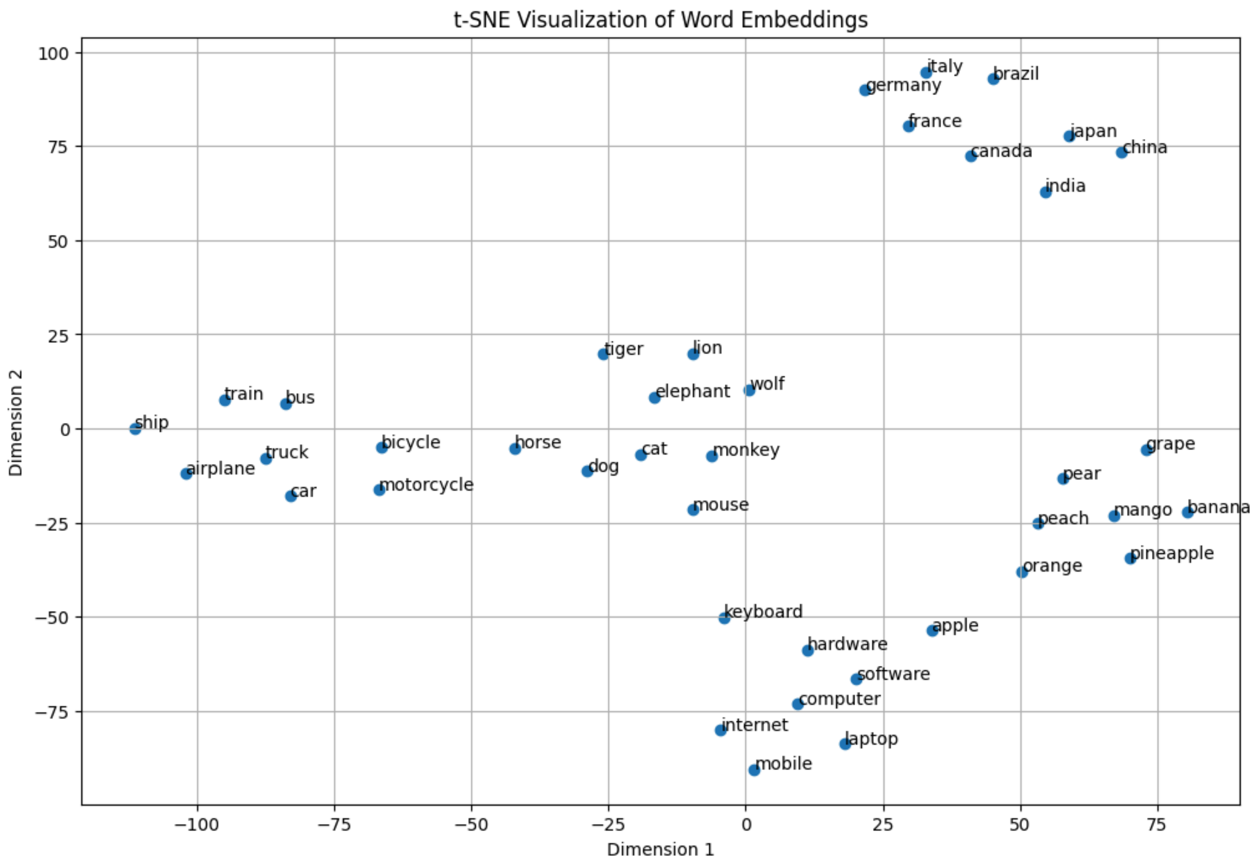
```
# Create a new figure for the plot with a specified size for better readability
plt.figure(figsize=(12, 8))

# Create a scatter plot of the reduced 2D word embeddings
plt.scatter(x_coords, y_coords)

# Annotate each point on the scatter plot with its corresponding word
for i, word in enumerate(words):
    plt.annotate(word, (x_coords[i], y_coords[i]))

# Set the title of the plot
plt.title("t-SNE Visualization of Word Embeddings")
# Label the x-axis
plt.xlabel("Dimension 1")
# Label the y-axis
plt.ylabel("Dimension 2")
# Add a grid to the plot for easier interpretation
plt.grid(True)

# Display the plot
plt.show()
```


t-SNE Visualization of Word Embeddings
Dimension 1
Dimension 2
italy
brazil
germany
france
japan
canada
china
india
tiger
lion
elephant
wolf
train
bus
ship
bicycle
horse
cat
monkey
dog
truck
airplane
car
motorcycle
mouse
grape
pear
mango
banana
peach
pineapple
orange
keyboard
apple
hardware
software
computer
internet
laptop
mobile