NAME : SAI CHARAN . P

ROLL:NO: 2403a52343

LAB : 10

Title: Visualizing Word Embeddings using t-SNE

STEP 1: Import Libraries

```
# Install gensim if not already installed
!pip install gensim

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

# For handling vectors and matrices
import numpy as np

# For dimensionality reduction (t-SNE)
from sklearn.manifold import TSNE

# For visualization
import matplotlib.pyplot as plt
```

```
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)
```

STEP 2: Load Embedding Model

```
model = api.load("glove-wiki-gigaword-100")

# Print vocabulary size
print("Vocabulary size:", len(model.key_to_index))

# Display one example vector
print("Vector for 'computer':")
print(model['computer'])
```

```
Vocabulary size: 400000
Vector for '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]
```

STEP 3: Select Word List

```
words = [
    # Animals
    "dog", "cat", "lion", "tiger", "elephant", "wolf",
    # Cities
    "paris", "london", "tokyo", "delhi", "berlin", "rome",
    # Technology
    "computer", "laptop", "internet", "software", "keyboard", "mouse",
    # Fruits
    "apple", "banana", "orange", "grape", "mango", "pineapple",
    # Vehicles
    "car", "bus", "train", "bicycle", "airplane", "truck"
]

# Extract vectors
```

```
vectors = np.array([model[word] for word in words])
print(vectors)
```

```
[[ 0.30817   0.30938   0.52803  ...  0.12883   0.62529  -0.52086 ]
 [ 0.23088   0.28283   0.6318   ... -0.14532   0.15104  -0.71493 ]
 [ 0.19713   0.43847   0.3363   ...  0.080824  0.76018  -0.47588 ]
 ...
 [ 0.54217  -0.05124  -0.45694  ... -0.12509   1.3863   -0.58022 ]
 [-0.16905   0.27387   0.56794  ...  0.010633  0.56534  -0.96648 ]
 [-0.13959   0.053049  0.098775 ...  0.77447   1.5429   -0.20753 ]]
```

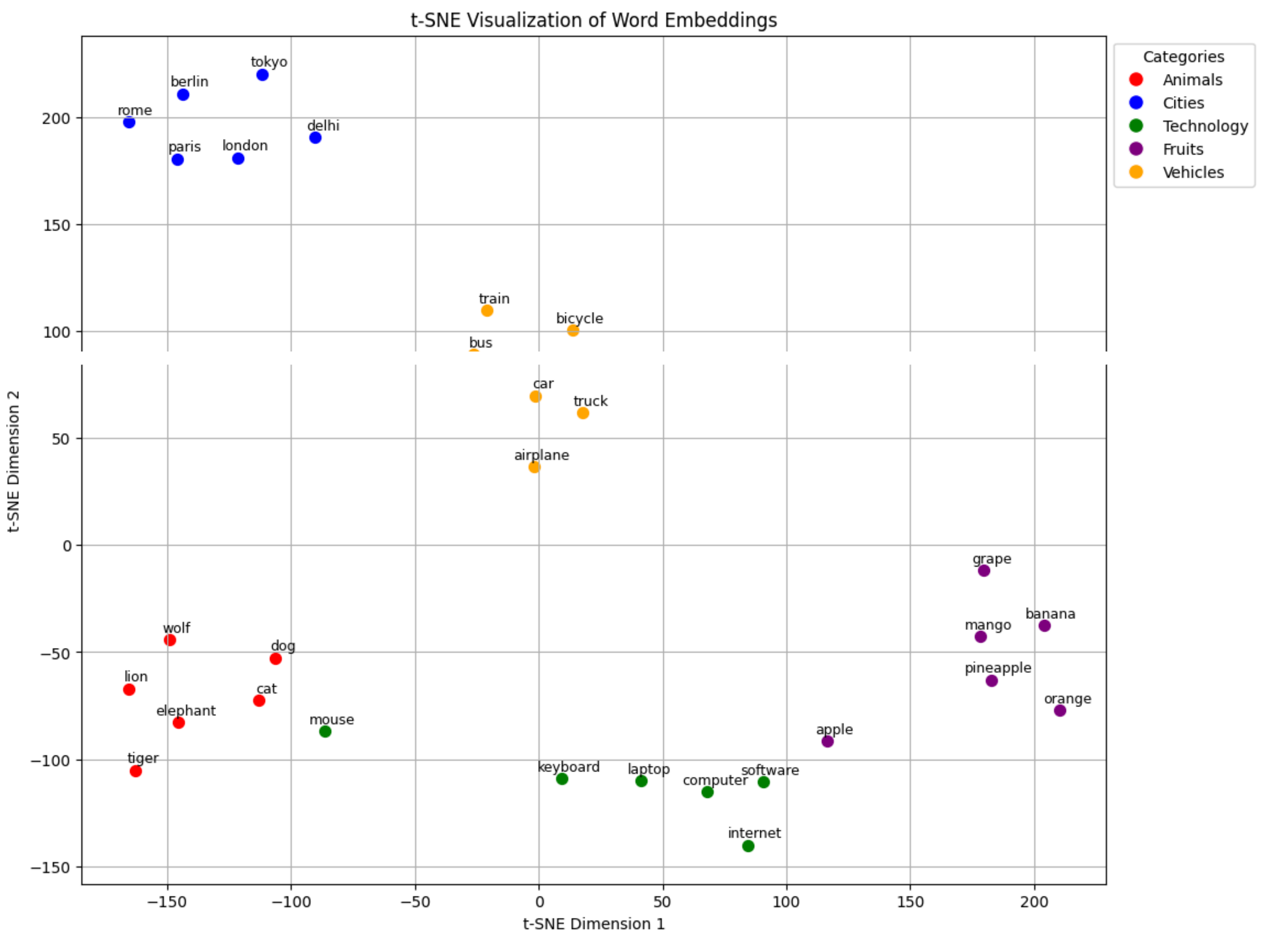
STEP 4: Apply t-SNE

```
from sklearn.manifold import TSNE

# Apply t-SNE
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
reduced_vectors = tsne.fit_transform(vectors)

# Print confirmation
print("t-SNE dimensionality reduction completed successfully.\n")

# Print shape of reduced vectors
print("Shape of reduced vectors:", reduced_vectors.shape, "\n")

# Print word with corresponding 2D coordinates
print("Word Coordinates after t-SNE:\n")
for i, word in enumerate(words):
    print(f"{word}: ({reduced_vectors[i][0]:.4f}, {reduced_vectors[i][1]:.4f})")
```

```
t-SNE dimensionality reduction completed successfully.

Shape of reduced vectors: (30, 2)

Word Coordinates after t-SNE:

dog: (-106.2158, -52.6437)
cat: (-113.0255, -72.3488)
lion: (-165.5599, -67.1036)
tiger: (-162.5691, -105.1715)
elephant: (-145.3049, -82.6403)
wolf: (-149.1827, -44.2536)
paris: (-145.7798, 180.5024)
london: (-121.4121, 181.0462)
tokyo: (-111.8421, 220.1769)
delhi: (-90.1837, 190.6520)
berlin: (-143.7824, 210.9476)
rome: (-165.6504, 197.6542)
computer: (68.1649, -115.3302)
laptop: (41.4986, -109.9356)
internet: (84.4897, -140.2469)
software: (90.5526, -110.4449)
keyboard: (9.3634, -108.9001)
mouse: (-86.3322, -86.8918)
apple: (116.5204, -91.5874)
banana: (204.0230, -37.6642)
orange: (210.3116, -77.2451)
grape: (179.7312, -11.8079)
mango: (178.4943, -42.6956)
pineapple: (182.7630, -62.9056)
car: (-1.4228, 69.8460)
bus: (-26.4314, 89.2540)
train: (-20.8945, 109.7729)
bicycle: (13.7959, 100.6645)
airplane: (-1.6529, 36.5276)
truck: (17.9767, 62.0239)
```

STEP 5: Plot Visualization

```
word_categories = [
    "Animals", "Animals", "Animals", "Animals", "Animals", "Animals",
    "Cities", "Cities", "Cities", "Cities", "Cities", "Cities",
    "Technology", "Technology", "Technology", "Technology", "Technology", "Technology",
    "Fruits", "Fruits", "Fruits", "Fruits", "Fruits", "Fruits",
    "Vehicles", "Vehicles", "Vehicles", "Vehicles", "Vehicles", "Vehicles"
]

category_colors = {
    "Animals": "red",
    "Cities": "blue",
    "Technology": "green",
    "Fruits": "purple",
    "Vehicles": "orange"
}

plt.figure(figsize=(12, 10))

# Plotting points and annotations
for i, word in enumerate(words):
    category = word_categories[i]
    color = category_colors[category]
    plt.scatter(reduced_vectors[i, 0], reduced_vectors[i, 1], c=color, s=50, label=category if word not in words[:i] else '
    plt.annotate(word, (reduced_vectors[i, 0], reduced_vectors[i, 1]), textcoords="offset points", xytext=(5,5), ha='center

# Create custom legend handles to avoid duplicate labels
legend_handles = []
for category, color in category_colors.items():
    legend_handles.append(plt.Line2D([0], [0], marker='o', color='w', label=category,
                                      markerfacecolor=color, markersize=10))

plt.legend(handles=legend_handles, title="Categories", loc='upper left', bbox_to_anchor=(1, 1))
plt.title('t-SNE Visualization of Word Embeddings')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.grid(True)
plt.show()
```



STEP 6: Interpretation

The visualization shows several meaningful clusters formed by semantically related words. Animal names such as dog, cat, lion, and tiger appear close to one another, forming a clear cluster. City names such as Paris, London, and Tokyo are grouped together in another region of the plot. Technology-related words like computer, laptop, and software also appear close, indicating similar contextual usage. Fruit names cluster together, suggesting shared semantic relationships. Vehicles such as car, bus, and train form another visible group. Some words may appear slightly distant from their category due to multiple meanings or contextual overlap. For example, “apple” may appear closer to technology words due to its association with the Apple company. Overall, the clusters demonstrate that embeddings capture semantic similarity effectively.

STEP 7: Lab Report Structure

A lab report for this analysis would typically include the following sections:

1. **Objective:** Clearly state the purpose of the experiment, which is to visualize word embeddings and understand semantic relationships using t-SNE.
2. **Methodology:** Detail the steps taken:
   - **Data Source:** Specify the pre-trained word embedding model used (e.g., GloVe-wiki-gigaword-100).
   - **Word Selection:** Describe the process of selecting a diverse list of words belonging to various semantic categories.
   - **Dimensionality Reduction:** Explain the application of t-SNE, including key parameters like `n_components` (2) and `perplexity` (5).
   - **Visualization:** Mention the use of `matplotlib` for creating the scatter plot and annotating words.
3. **Results:** Present the t-SNE visualization and discuss observations:
   - **Clustering:** Describe how words belonging to similar semantic categories (e.g., animals, cities, technology) form distinct clusters.
   - **Semantic Proximity:** Explain that the proximity of words in the 2D space reflects their semantic similarity in the original high-dimensional embedding space.
   - **Outliers/Nuances:** Briefly discuss any words that might appear slightly out of place from their expected clusters, possibly due to contextual nuances in the training data.
4. **Conclusion:** Summarize the key findings and their implications:
   - Reiterate that word embeddings effectively capture semantic relationships.
   - Conclude that t-SNE is a valuable tool for visualizing and interpreting these high-dimensional embeddings.

This structure provides a comprehensive overview of the experiment, from its goal to its interpretation.