NAME : NITHIN.V

ROLL.NO : 2403A52355

BATCH.NO : 13(AIML)

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
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8.4 kB)
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)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 64.9 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

STEP 2 — Load Embedding Model

```
# Load GloVe pre-trained embeddings (100 dimensional)
model = api.load("glove-wiki-gigaword-100")
# Print vocabulary size
print("Vocabulary Size:", len(model.key_to_index))
# Display one example word vector
example_word = "computer"
print("\nExample Word:", example_word)
print("Vector Shape:", model[example_word].shape)
print("First 10 values of vector:\n", model[example_word][:10])
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
Vocabulary Size: 400000

Example Word: computer
Vector Shape: (100,)
First 10 values of vector:
 [-0.16298   0.30141   0.57978   0.066548  0.45835  -0.15329   0.43258
 -0.89215   0.57747   0.36375 ]
```

STEP 3 — Select Word List

```
# Create grouped word list (40 words)
word_list = [
    # Animals
    "dog", "cat", "lion", "tiger", "elephant", "horse", "cow", "goat",
    # Cities
    "paris", "london", "tokyo", "delhi", "mumbai", "newyork", "rome",
    # Countries
    "india", "france", "japan", "china", "germany", "brazil",
    # Technology
    "computer", "laptop", "keyboard", "mouse", "internet",
    "software", "hardware", "mobile",
    # Fruits
    "apple", "banana", "mango", "orange", "grape", "pineapple"
]
# Extract vectors for selected words
vectors = []
valid_words = []  # store words found in vocabulary
for word in word_list:
    if word in model:
        vectors.append(model[word])
        valid_words.append(word)
vectors = np.array(vectors)
```

```
print("Number of words used:", len(valid_words))
print("Vector matrix shape:", vectors.shape)
```

```
Number of words used: 35
Vector matrix shape: (35, 100)
```

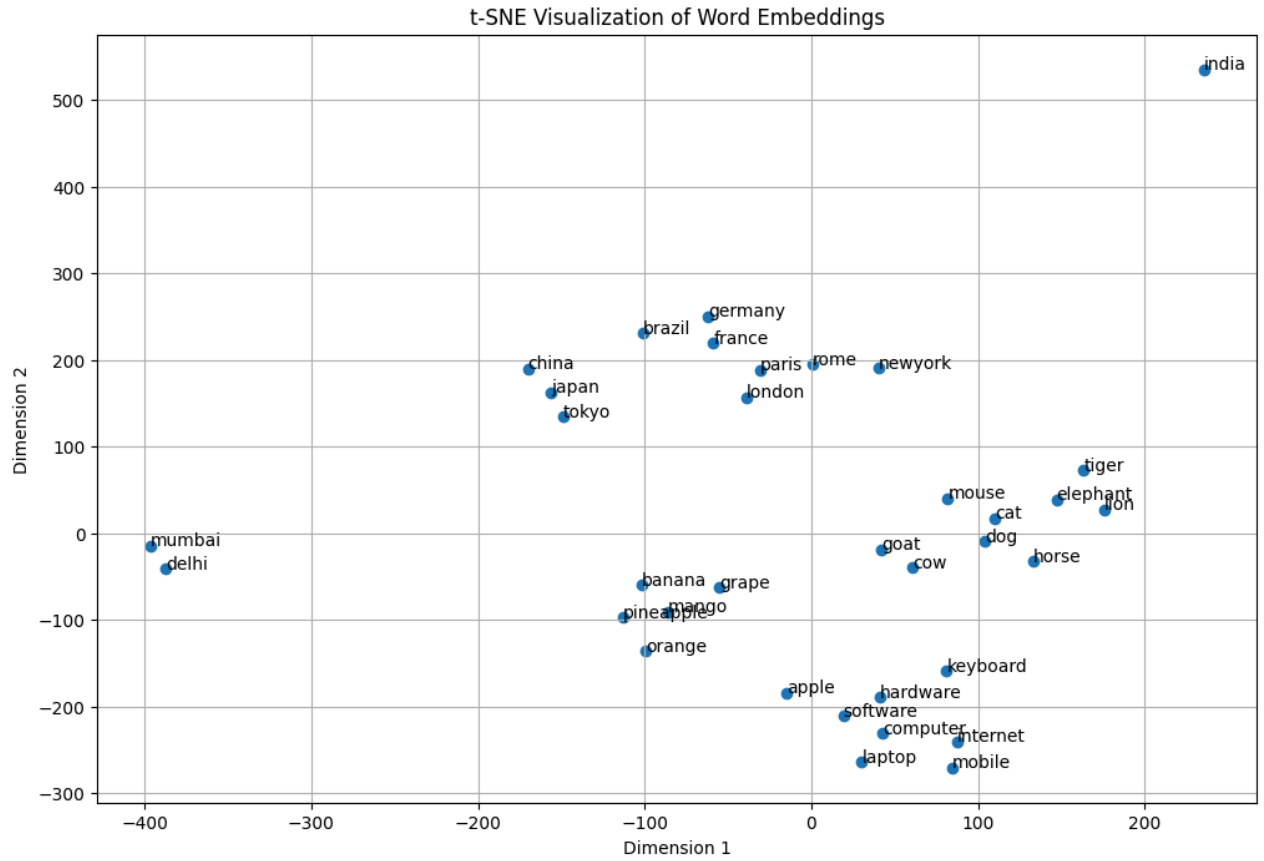
STEP 4 — Apply t-SNE

```
# Reduce from 100 dimensions → 2 dimensions
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
reduced_vectors = tsne.fit_transform(vectors)
print("Reduced shape:", reduced_vectors.shape)
```

```
Reduced shape: (35, 2)
```

STEP 5 — Plot Visualization

```
plt.figure(figsize=(12, 8))
# Scatter plot
plt.scatter(reduced_vectors[:, 0], reduced_vectors[:, 1])
# Annotate each word
for i, word in enumerate(valid_words):
    plt.annotate(word,
                 (reduced_vectors[i, 0], reduced_vectors[i, 1]))
plt.title("t-SNE Visualization of Word Embeddings")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.grid(True)
plt.show()
```



STEP 6 — Interpretation

The t-SNE visualization shows clear clustering among semantically related words. Animal words such as dog, cat, lion, and tiger appear close to each other, forming a visible cluster. City names like Paris, London, and Tokyo are grouped together, indicating geographical similarity. Country names such as India, France, and Japan also appear relatively close. Technology-related words including computer, laptop, and software form another cluster. Fruit names like apple, banana, and mango are positioned near each other. Words belonging to similar categories are placed closer because their embeddings are similar in high-dimensional space. Some words may appear

slightly away from their group due to contextual usage differences in training data. Overall, the visualization confirms that word embeddings capture semantic relationships 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.