```
!pip install gensim
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import gensim.downloader as api
```

```
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 49.0 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

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

print("Vocabulary size:", len(model.index_to_key))

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

```
[==================================================] 100.0% 128.1/128.1MB downloaded
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]
```

```
# Load dataset
data = pd.read_csv("word_list_tsne.csv")
words = data["word"].tolist()

print("Total words:", len(words))
print(words)
```

```
Total words: 39
['dog', 'cat', 'lion', 'tiger', 'elephant', 'horse', 'cow', 'monkey', 'apple', 'banana', 'orange', 'mango', 'grape', 'pineappl
```

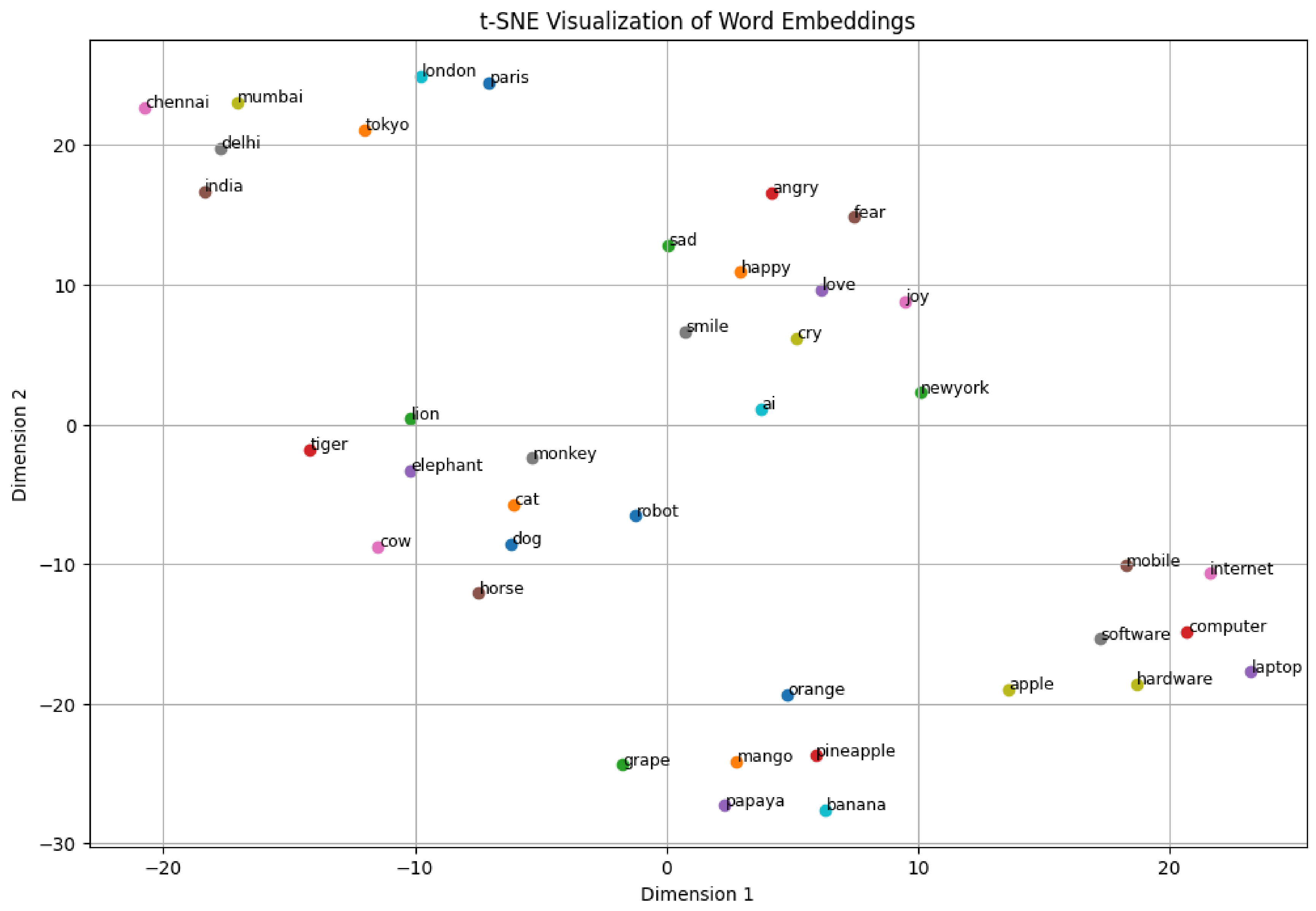
```
word_vectors = []
valid_words = []

for word in words:
    if word in model:
        valid_words.append(word)
        word_vectors.append(model[word])

word_vectors = np.array(word_vectors)

print("Valid words:", len(valid_words))
```

```
Valid words: 39
```

```
tsne = TSNE(n_components=2, random_state=42, perplexity=10)
reduced_vectors = tsne.fit_transform(word_vectors)

print("Shape after t-SNE:", reduced_vectors.shape)
```

```
Shape after t-SNE: (39, 2)
```

```
plt.figure(figsize=(12, 8))

for i, word in enumerate(valid_words):
    x, y = reduced_vectors[i]
    plt.scatter(x, y)
    plt.text(x+0.02, y+0.02, word, fontsize=9)

plt.title("t-SNE Visualization of Word Embeddings")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.grid(True)
plt.show()
```



INTERPRETATION: The t-SNE visualization shows that words with similar meanings are grouped together. Animal-related words such as dog, cat, lion, and tiger form a cluster, indicating semantic similarity. Fruit-related words like apple, banana, mango, and orange also appear close to each other. Technology-related words such as computer, laptop, mobile, and software form another cluster. City names like chennai, delhi, mumbai, london, and paris appear in nearby regions. Emotion-related words such as happy, joy, love, and smile are grouped together, while sad and angry appear slightly apart. Some words may appear closer than expected due to contextual similarities in the training corpus. This visualization demonstrates how word embeddings capture semantic relationships between words. Overall, t-SNE effectively helps in understanding the structure of high-dimensional word embeddings.

Close

```
LAB REPORT:

Objective

To visualize high-dimensional word embeddings using t-SNE and
analyze semantic relationships between words.

📌 Embedding Model Description

We used the GloVe pre-trained word embedding model with
100-dimensional vectors. GloVe captures semantic meaning by
analyzing word co-occurrence patterns in large text corpora.

📌 Word List Used

Animals, fruits, cities, technology terms, and emotions (40
words).

📌 Result
```

LAB REPORT:

Objective

To visualize high-dimensional word embeddings using t-SNE and analyze semantic relationships between words.

📌 Embedding Model Description

We used the GloVe pre-trained word embedding model with 100-dimensional vectors. GloVe captures semantic meaning by analyzing word co-occurrence patterns in large text corpora.

📌 Word List Used

Animals, fruits, cities, technology terms, and emotions (40 words).

📌 Result

t-SNE reduced 100-dimensional vectors into 2D space and