Step 1 : Open colab

STEP 2 — Import Libraries

```
!pip install gensim
import gensim.downloader as api          # to load pre-trained embeddings
import numpy as np                       # for numerical matrix operations
import matplotlib.pyplot as plt          # for visualization (scatter plot)
from sklearn.manifold import TSNE        # for dimensionality reduction
```

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

STEP 3 — Load Embedding Model

```
# Load pre-trained Word2Vec model (Google News 300D)
model = api.load("word2vec-google-news-300")

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

# Display one example vector
print("Vector for word 'king':\n", model['king'])
print("Vector dimension:", len(model['king']))
```

```
 -1.77734375e-01  8.59375000e-02 -2.18505859e-02  2.05078125e-02
 -1.39648438e-01  2.51464844e-02  1.38671875e-01 -1.05468750e-01
  1.38671875e-01  8.88671875e-02 -7.51953125e-02 -2.13623047e-02
  1.72851562e-01  4.63867188e-02 -2.65625000e-01  8.91113281e-03
  1.49414062e-01  3.78417969e-02  2.38281250e-01 -1.24511719e-01
 -2.17773438e-01 -1.81640625e-01  2.97851562e-02  5.71289062e-02
 -2.89306641e-02  1.24511719e-02  9.66796875e-02 -2.31445312e-01
  5.81054688e-02  6.68945312e-02  7.08007812e-02 -3.08593750e-01
 -2.14843750e-01  1.45507812e-01 -4.27734375e-01 -9.39941406e-03
  1.54296875e-01 -7.66601562e-02  2.89062500e-01  2.77343750e-01
 -4.86373901e-04 -1.36718750e-01  3.24218750e-01 -2.46093750e-01
 -3.03649902e-03 -2.11914062e-01  1.25000000e-01  2.69531250e-01
  2.04101562e-01  8.25195312e-02 -2.01171875e-01 -1.60156250e-01
 -3.78417969e-02 -1.20117188e-01  1.15234375e-01 -4.10156250e-02
 -3.95507812e-02 -8.98437500e-02  6.34765625e-03  2.03125000e-01
  1.86523438e-01  2.73437500e-01  6.29882812e-02  1.41601562e-01
 -9.81445312e-02  1.38671875e-01  1.82617188e-01  1.73828125e-01
  1.73828125e-01 -2.37304688e-01  1.78710938e-01  6.34765625e-02
  2.36328125e-01 -2.08984375e-01  8.74023438e-02 -1.66015625e-01
 -7.91015625e-02  2.43164062e-01 -8.88671875e-02  1.26953125e-01
 -2.16796875e-01 -1.73828125e-01 -3.59375000e-01 -8.25195312e-02
 -6.49414062e-02  5.07812500e-02  1.35742188e-01 -7.47070312e-02
```

```
  1.57226562e-01 -1.14257812e-01 -2.05078125e-01  9.91210938e-02
  3.69140625e-01 -1.97265625e-01  3.54003906e-02  1.09375000e-01
  1.31835938e-01  1.66992188e-01  2.35351562e-01  1.04980469e-01
 -4.96093750e-01 -1.64062500e-01 -1.56250000e-01 -5.22460938e-02
  1.03027344e-01  2.43164062e-01 -1.88476562e-01  5.07812500e-02
 -9.37500000e-02 -6.68945312e-02  2.27050781e-02  7.61718750e-02
  2.89062500e-01  3.10546875e-01 -5.37109375e-02  2.28515625e-01
  2.51464844e-02  6.78710938e-02 -1.21093750e-01 -2.15820312e-01
 -2.73437500e-01 -3.07617188e-02 -3.37890625e-01  1.53320312e-01
  2.33398438e-01 -2.08007812e-01  3.73046875e-01  8.20312500e-02
  2.51953125e-01 -7.61718750e-02 -4.66308594e-02 -2.23388672e-02
  2.99072266e-02 -5.93261719e-02 -4.66918945e-03 -2.44140625e-01
 -2.09960938e-01 -2.87109375e-01 -4.54101562e-02 -1.77734375e-01
 -2.79296875e-01 -8.59375000e-02  9.13085938e-02  2.51953125e-01]
Vector dimension: 300
```

STEP 4 — Select Word List

```
words = [
    # Animals
    "cat", "dog", "lion", "tiger", "elephant", "wolf", "horse",

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

    # Countries
    "india", "china", "france", "germany", "japan", "america",

    # Cities
    "delhi", "mumbai", "paris", "tokyo", "berlin",

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

    # Royalty
    "king", "queen", "prince", "princess"
]
```

```
vectors = []

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

vectors = np.array(vectors)
print("Shape of vector matrix:", vectors.shape)
```

```
Shape of vector matrix: (34, 300)
```

STEP 5 — Apply t-SNE

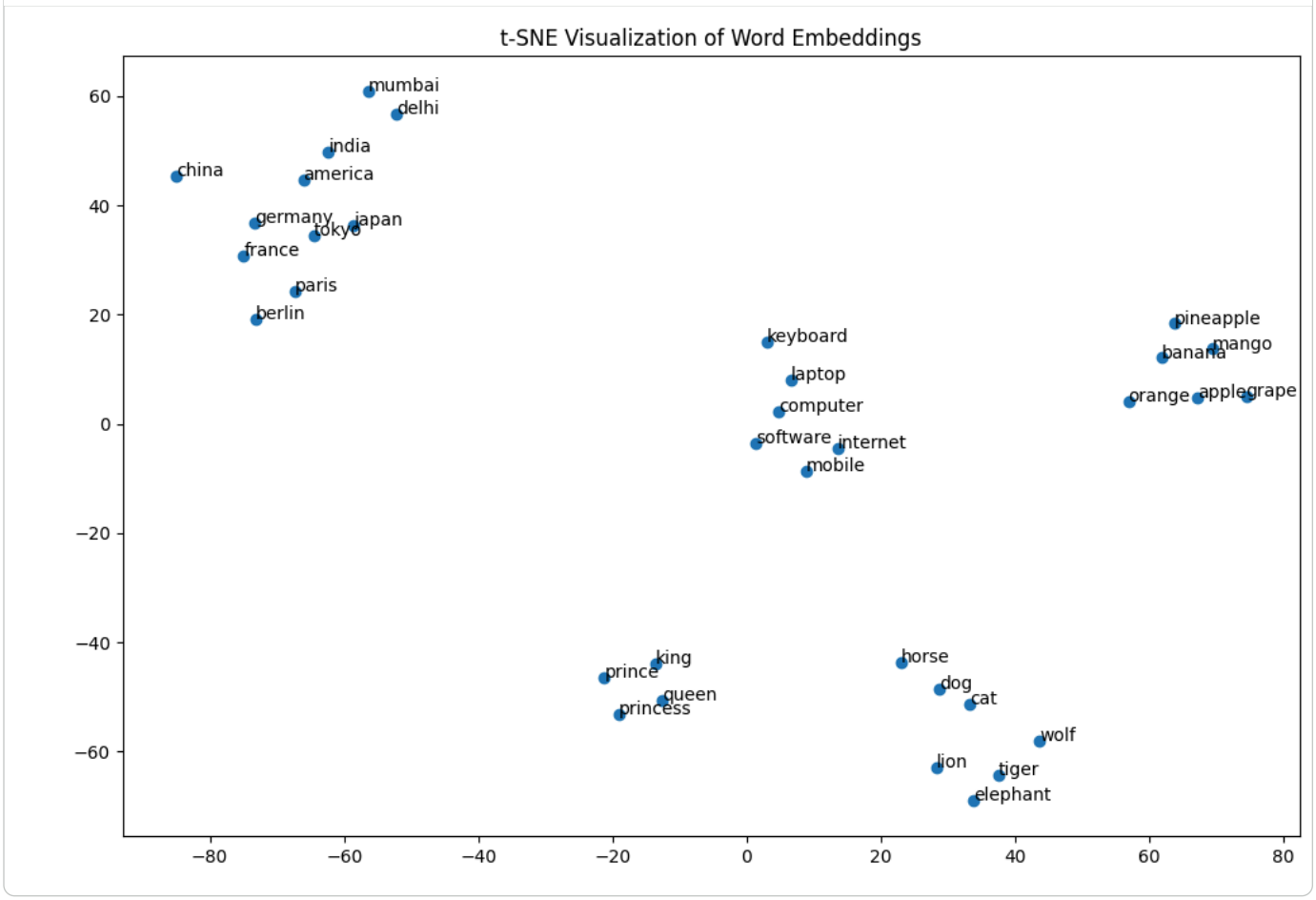
```
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
reduced_vectors = tsne.fit_transform(vectors)

print("Reduced shape:", reduced_vectors.shape)
```

```
Reduced shape: (34, 2)
```

STEP 6 — Plot Visualization

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

x = reduced_vectors[:,0]
y = reduced_vectors[:,1]

plt.scatter(x, y)

# Annotate words
for i, word in enumerate(words):
    plt.annotate(word, (x[i], y[i]))

plt.title("t-SNE Visualization of Word Embeddings")
```

```
plt.show()
```


t-SNE Visualization of Word Embeddings
mumbai
delhi
india
china
america
germany
japan
tokyo
france
paris
berlin
keyboard
laptop
computer
software
internet
mobile
pineapple
mango
banana
orange
apple
grape
king
prince
queen
princess
horse
dog
cat
wolf
lion
tiger
elephant
60
40
20
0
−20
−40
−60
−80
−60
−40
−20
0
20
40
60
80