

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

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

# For performing numerical operations, especially with arrays and matrices
import numpy as np

# For creating static, interactive, and animated visualizations in Python
import matplotlib.pyplot as plt

# For dimensionality reduction, specifically t-distributed Stochastic Neighbor Embedding
from sklearn.manifold import TSNE
```

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 50.2 MB/s eta 0:00:00
```

```
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

```
# Load a pre-trained GloVe model with 100 dimensions from the gensim API
# This model was trained on Wikipedia 2014 + Gigaword 5 text data
model = api.load("glove-wiki-gigaword-100")

# Print the total number of unique words present in the loaded model's vocabulary
print("Vocabulary size:", len(model.key_to_index))

# Display the word vector (embedding) for the word 'computer'
print("\nVector for word 'computer':\n")
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]
```

```
# Define a list of words categorized by different themes (animals, fruits, countries, technology, vehicles, emotions)
# These words will be used to demonstrate word embedding visualization
words = [
    # Animals
    "dog", "cat", "lion", "tiger", "elephant", "horse", "cow", "goat",

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

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

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

    # Vehicles
    "car", "bus", "train", "airplane", "bicycle", "motorcycle",
```

```
# Emotions
"happy", "sad", "angry", "fear", "love", "joy"
]

# Extract the 100-dimensional word vectors for each word in the 'words' list
# These vectors are then stacked into a NumPy array
vectors = np.array([model[word] for word in words])
```

```
# Print the list of selected words that were used for extracting vectors
print("Selected Words:\n")
print(words)

# Print the total count of words selected
print("\nTotal words selected:", len(words))

# Print the shape of the 'vectors' NumPy array. It should be (number of words, embedding dimension)
print("\nShape of extracted vectors:", vectors.shape)

# Print the first 5 values of the word vector for the first word in the 'words' list (which is 'dog')
print("\nFirst 5 values of vector for word:", words[0])
print(vectors[0][:5])
```

Selected Words:

```
['dog', 'cat', 'lion', 'tiger', 'elephant', 'horse', 'cow', 'goat', 'apple', 'banana', 'mango', 'orange', 'grape', 'pineapple',
```

Total words selected: 40

Shape of extracted vectors: (40, 100)

```
First 5 values of vector for word: dog
[ 0.30817  0.30938  0.52803 -0.92543 -0.73671]
```

```
# Initialize a t-SNE model for dimensionality reduction
# n_components=2 means we want to reduce the vectors to 2 dimensions for visualization
# random_state=42 ensures reproducibility of the results
# perplexity=10 is a common hyperparameter for t-SNE, influencing how local and global aspects are balanced
tsne = TSNE(n_components=2, random_state=42, perplexity=10)

# Apply t-SNE to the high-dimensional word vectors to reduce them to 2 dimensions
reduced_vectors = tsne.fit_transform(vectors)

# Print the shape of the 'reduced_vectors' array, which should now be (number of words, 2)
print("Shape after reduction:", reduced_vectors.shape)
```

Shape after reduction: (40, 2)

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

# Extract the x-coordinates (first dimension) from the reduced vectors
x = reduced_vectors[:,0]
# Extract the y-coordinates (second dimension) from the reduced vectors
y = reduced_vectors[:,1]

# Create a scatter plot using the 2D reduced vectors
plt.scatter(x, y)

# Annotate each point in the scatter plot with its corresponding word
# This loop iterates through the index and word for each item in the 'words' list
for i, word in enumerate(words):
    # Place the word annotation at the (x[i], y[i]) coordinates
    plt.annotate(word, (x[i], y[i]))

# Set the title of the plot
plt.title("t-SNE Visualization of Word Embeddings")
# Set the label for the x-axis
plt.xlabel("Dimension 1")
# Set the label for the y-axis
plt.ylabel("Dimension 2")

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

