```
!pip install gensim
print("gensim installed successfully.")
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 64.7 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
gensim installed successfully.
```

```
import gensim.downloader as api
print("gensim.downloader imported for loading pre-trained word embedding models."

import numpy as np
print("numpy imported for numerical operations and handling word vectors.")

import pandas as pd
print("pandas imported for data manipulation and analysis.")

import matplotlib.pyplot as plt
print("matplotlib.pyplot imported for creating visualizations.")

from sklearn.manifold import TSNE
print("TSNE from sklearn.manifold imported for dimensionality reduction.")
```

```
gensim.downloader imported for loading pre-trained word embedding models.
numpy imported for numerical operations and handling word vectors.
pandas imported for data manipulation and analysis.
matplotlib.pyplot imported for creating visualizations.
TSNE from sklearn.manifold imported for dimensionality reduction.
```

```
print("Loading pre-trained GloVe model (glove-wiki-gigaword-100). This may take
model = api.load("glove-wiki-gigaword-100")
print("Model loaded successfully.")

vocabulary_size = len(model.index_to_key)
print(f"\nVocabulary size of the loaded model: {vocabulary_size} words")

example_word = 'computer'
if example_word in model:
    example_vector = model[example_word]
    print(f"\nVector for the word '{example_word}':\n{example_vector}")
    print("\nThis vector is a numerical representation of the word 'computer'."
    print("Words with similar meanings are expected to have similar vectors in
else:
    print(f"\nThe word '{example_word}' is not in the model's vocabulary.")
```

```
Loading pre-trained GloVe model (glove-wiki-gigaword-100). This may take a few m
Model loaded successfully.
```

```
Vocabulary size of the loaded model: 400000 words

Vector for the 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]

This vector is a numerical representation of the word 'computer'.
Words with similar meanings are expected to have similar vectors in this high-di
```

```
selected_words = [
    # Animals
    'cat', 'dog', 'lion', 'tiger', 'elephant', 'zebra', 'mouse', 'bird',
    # Food
    'apple', 'banana', 'orange', 'grape', 'strawberry', 'pizza', 'pasta', 'sushi'
    # Technology
    'computer', 'software', 'internet', 'phone', 'keyboard', 'monitor', 'algorith
    # Emotions
    'happy', 'sad', 'angry', 'joy', 'love', 'fear', 'excitement', 'calm',
    # Countries
    'usa', 'canada', 'mexico', 'brazil', 'france', 'germany', 'japan', 'china'
]

print(f"Selected {len(selected_words)} words for analysis.")
```

```
Selected 40 words for analysis.
```

```
word_vectors = []
actual_words = []

for word in selected_words:
    if word in model:
        word_vectors.append(model[word])
        actual_words.append(word)
    else:
        print(f"Warning: Word '{word}' not found in the model's vocabulary. Ski

word_vectors_array = np.array(word_vectors)

print(f"\nExtracted vectors for {len(actual_words)} out of {len(selected_words)
print(f"Shape of word_vectors_array: {word_vectors_array.shape}")
```

```
Extracted vectors for 40 out of 40 selected words.
Shape of word_vectors_array: (40, 100)
```

```
tsne = TSNE(n_components=2, random_state=42, perplexity=5, n_iter=2500, learning_
tsne_results = tsne.fit_transform(word_vectors_array)

print("t-SNE dimensionality reduction completed.")
print(f"Shape of 2D t-SNE results: {tsne_results.shape}")
```

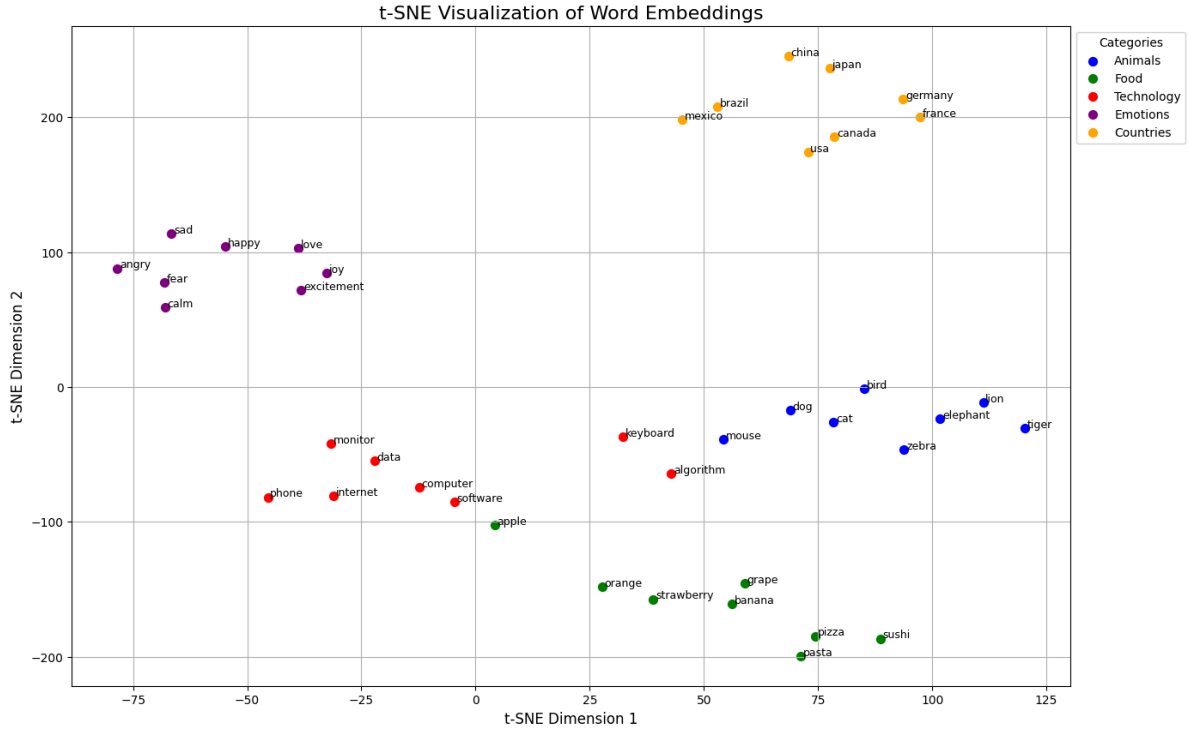
```
/usr/local/lib/python3.12/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureW
  warnings.warn(
t-SNE dimensionality reduction completed.
Shape of 2D t-SNE results: (40, 2)
```

```
tsne = TSNE(n_components=2, random_state=42, perplexity=5, max_iter=2500, learn
tsne_results = tsne.fit_transform(word_vectors_array)

print("t-SNE dimensionality reduction completed.")
print(f"Shape of 2D t-SNE results: {tsne_results.shape}")
```

```
t-SNE dimensionality reduction completed.
Shape of 2D t-SNE results: (40, 2)
```

```
category_map = {
    'Animals': ['cat', 'dog', 'lion', 'tiger', 'elephant', 'zebra', 'mouse', 'b
    'Food': ['apple', 'banana', 'orange', 'grape', 'strawberry', 'pizza', 'past
    'Technology': ['computer', 'software', 'internet', 'phone', 'keyboard', 'mo
    'Emotions': ['happy', 'sad', 'angry', 'joy', 'love', 'fear', 'excitement',
    'Countries': ['usa', 'canada', 'mexico', 'brazil', 'france', 'germany', 'ja
}

word_to_category = {}
for category, words in category_map.items():
    for word in words:
        if word in actual_words: # Ensure only words present in actual_words ar
            word_to_category[word] = category

# Define colors for each category
colors = {
    'Animals': 'blue',
    'Food': 'green',
    'Technology': 'red',
    'Emotions': 'purple',
    'Countries': 'orange'
}

plt.figure(figsize=(15, 10))

# Separate data by category for plotting
category_data = {cat: {'x': [], 'y': [], 'words': []} for cat in colors.keys()}

for i, word in enumerate(actual_words):
    category = word_to_category.get(word, 'Unknown')
    if category == 'Unknown': # Handle words not explicitly categorized
```

```
            if 'Unknown' not in category_data: # Add 'Unknown' category if it appea
                category_data['Unknown'] = {'x': [], 'y': [], 'words': []}
                colors['Unknown'] = 'grey'

        category_data[category]['x'].append(tsne_results[i, 0])
        category_data[category]['y'].append(tsne_results[i, 1])
        category_data[category]['words'].append(word)

    # Plot each category
    for category, data in category_data.items():
        if data['x']:
            plt.scatter(data['x'], data['y'], color=colors[category], label=categor
            for j, word in enumerate(data['words']):
                plt.annotate(word, (data['x'][j] + 0.5, data['y'][j] + 0.5), fontsi

    plt.title('t-SNE Visualization of Word Embeddings', fontsize=16)
    plt.xlabel('t-SNE Dimension 1', fontsize=12)
    plt.ylabel('t-SNE Dimension 2', fontsize=12)
    plt.legend(title='Categories', loc='best', bbox_to_anchor=(1, 1))
    plt.grid(True)
    plt.show()
    print("t-SNE visualization plot generated with annotations and categorized poin
```



t-SNE visualization plot generated with annotations and categorized points.