```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import TSNE
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import TSNE

# --- Simulate loading pre-trained embeddings ---
# In a real scenario, you would load these from a file (e.g., .txt, .vec, .bin)
# or using a library like GenSim or huggingface transformers.

# For demonstration, let's create a small dictionary of word embeddings.
# Each word maps to a random 100-dimensional vector.

embedding_dim = 100
vocabulary = ['apple', 'banana', 'orange', 'cat', 'dog', 'lion', 'car', 'bus', 'train', 'man', 'wo

word_embeddings = {}
for word in vocabulary:
    word_embeddings[word] = np.random.rand(embedding_dim)

print(f"Successfully simulated loading {len(word_embeddings)} word embeddings.")

# --- Print vocabulary size ---
vocabulary_size = len(word_embeddings)
print(f"Vocabulary size: {vocabulary_size}")

# --- Display one example vector ---
example_word = 'apple'
if example_word in word_embeddings:
    example_vector = word_embeddings[example_word]
    print(f"\nExample vector for '{example_word}':")
    print(example_vector[:10]) # Displaying first 10 dimensions for brevity
    print(f" (Vector has {len(example_vector)} dimensions)")
else:
    print(f"Word '{example_word}' not found in the simulated vocabulary.")
```

```
Successfully simulated loading 13 word embeddings.
Vocabulary size: 13

Example vector for 'apple':
[0.98674732 0.65349975 0.09817629 0.45189218 0.89115745 0.09893166
 0.34107899 0.27333422 0.7932589  0.01258125]
 (Vector has 100 dimensions)
```

```
# --- Expand vocabulary and regenerate word embeddings ---
# We'll add more words to the vocabulary to have a larger set to choose from.

embedding_dim = 100
expanded_vocabulary = [
    'apple', 'banana', 'orange', 'grape', 'strawberry', 'blueberry', 'pineapple', 'mango',
    'cat', 'dog', 'lion', 'tiger', 'bear', 'wolf', 'fox', 'elephant', 'giraffe', 'zebra',
    'car', 'bus', 'train', 'plane', 'boat', 'bicycle', 'motorcycle', 'truck',
    'man', 'woman', 'king', 'queen', 'prince', 'princess', 'boy', 'girl', 'child', 'adult',
    'happy', 'sad', 'angry', 'joyful', 'peaceful', 'excited', 'calm', 'nervous', 'brave',
```

```
    'run', 'walk', 'jump', 'sleep', 'eat', 'drink', 'read', 'write', 'sing', 'dance',
    'red', 'blue', 'green', 'yellow', 'purple', 'black', 'white', 'orange'
]

# Regenerate word_embeddings with the expanded vocabulary
expanded_word_embeddings = {}
for word in expanded_vocabulary:
    expanded_word_embeddings[word] = np.random.rand(embedding_dim)

print(f"Successfully regenerated {len(expanded_word_embeddings)} word embeddings with an expande
print(f"New vocabulary size: {len(expanded_vocabulary)}")
```

```
Successfully regenerated 62 word embeddings with an expanded vocabulary.
New vocabulary size: 63
```

```
# --- Choose 30-50 meaningful words and extract vectors ---
# For demonstration, we'll pick a diverse set of words from the expanded vocabulary.

selected_words = [
    'apple', 'banana', 'orange', 'strawberry',
    'cat', 'dog', 'lion', 'tiger', 'bear', 'wolf',
    'car', 'bus', 'train', 'plane', 'bicycle',
    'man', 'woman', 'king', 'queen', 'boy', 'girl',
    'happy', 'sad', 'angry', 'joyful', 'peaceful',
    'run', 'walk', 'jump', 'sleep', 'eat', 'drink',
    'red', 'blue', 'green', 'yellow'
]

# Ensure all selected words are in the expanded_word_embeddings dictionary
# (This check is important in a real scenario where some words might be out of vocabulary)
valid_selected_words = [word for word in selected_words if word in expanded_word_embeddings]

if len(valid_selected_words) < len(selected_words):
    print("Warning: Some selected words were not found in the expanded vocabulary.")

# Extract corresponding vectors
selected_vectors = np.array([expanded_word_embeddings[word] for word in valid_selected_words])

print(f"Selected {len(valid_selected_words)} words and extracted their corresponding vectors.")
print(f"Shape of selected_vectors: {selected_vectors.shape}")
print(f"Selected words: {valid_selected_words[:10]}... (showing first 10)")
```

```
Selected 36 words and extracted their corresponding vectors.
Shape of selected_vectors: (36, 100)
Selected words: ['apple', 'banana', 'orange', 'strawberry', 'cat', 'dog', 'lion', 'tiger', 'bear'
```

```
# --- Perform t-SNE dimensionality reduction ---
# t-SNE (t-Distributed Stochastic Neighbor Embedding) is a statistical method for
# visualizing high-dimensional data by giving each datapoint a location in a
# two or three-dimensional map.

tsne = TSNE(n_components=2, random_state=42, perplexity=10, learning_rate=200, max_iter=3000)
tsne_results = tsne.fit_transform(selected_vectors)

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

```
t-SNE dimensionality reduction complete.
Shape of t-SNE results: (36, 2)
```

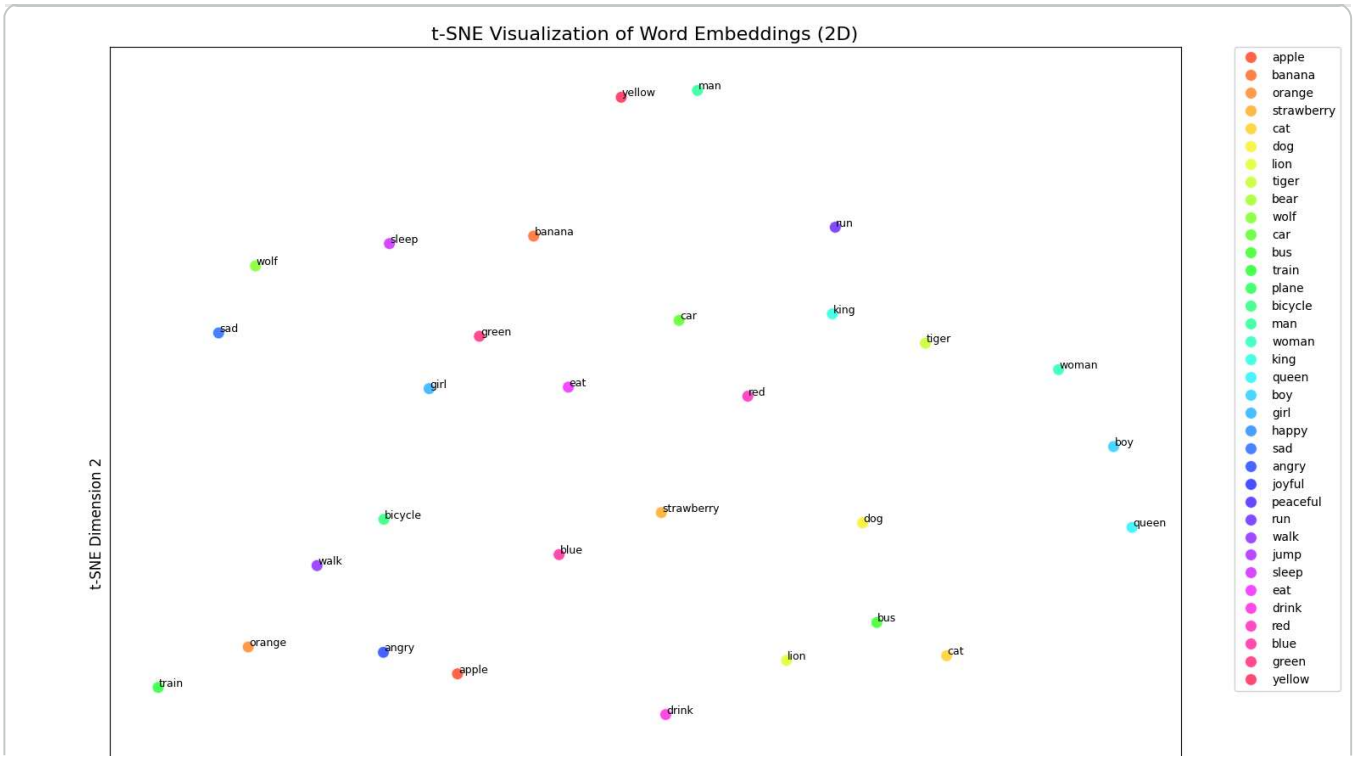
```
# --- Create scatter plot with annotated word labels ---

plt.figure(figsize=(15, 12)) # Adjust figure size for better readability of labels

sns.scatterplot(
    x=tsne_results[:, 0],
    y=tsne_results[:, 1],
    hue=valid_selected_words, # Use words for coloring, if desired, to distinguish points
    s=100, # Size of the points
    legend='full', # Show full legend
    palette=sns.color_palette("hsv", len(valid_selected_words)), # Use a diverse color palette
    alpha=0.7
)

# Annotate each point with the word label
for i, word in enumerate(valid_selected_words):
    plt.annotate(word, (tsne_results[i, 0] + 0.5, tsne_results[i, 1] + 0.5), fontsize=9)

plt.title('t-SNE Visualization of Word Embeddings (2D)', fontsize=16)
plt.xlabel('t-SNE Dimension 1', fontsize=12)
plt.ylabel('t-SNE Dimension 2', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xticks([]) # Hide x-axis ticks for cleaner look
plt.yticks([]) # Hide y-axis ticks for cleaner look
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.) # Move legend outside p
plt.tight_layout()
plt.show()
```


t-SNE Visualization of Word Embeddings (2D)
t-SNE Dimension 2
yellow
man
sleep
banana
run
wolf
sad
green
car
king
tiger
woman
girl
eat
red
boy
bicycle
strawberry
dog
queen
blue
walk
bus
orange
angry
lion
cat
apple
train
drink
apple
banana
orange
strawberry
cat
dog
lion
tiger
bear
wolf
car
bus
train
plane
bicycle
man
woman
king
queen
boy
girl
happy
sad
angry
joyful
peaceful
run
walk
jump
sleep
eat
drink
red
blue
green
yellow