```
Start coding or generate with AI.
```

```
import sys
!{sys.executable} -m 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_64.whl.metadata (8
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gens
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensi
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=
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 58.8 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
gensim installed successfully.
```

```
import gensim.models.keyedvectors as KeyedVectors
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import pandas as pd

print("Libraries imported successfully.")
```

```
Libraries imported successfully.
```

```
import gensim.downloader as api

# Load a pre-trained word embedding model
print("Downloading and loading 'glove-wiki-gigaword-50' model...")
model = api.load('glove-wiki-gigaword-50')
print("Model loaded successfully.")
```

```
Downloading and loading 'glove-wiki-gigaword-50' model...
[==================================================] 100.0% 66.0/66.0MB downloaded
Model loaded successfully.
```

```
print(f"Vocabulary size: {len(model.key_to_index)} words")

# Choose a common word
word = 'computer'

# Check if the word exists in the model's vocabulary
if word in model.key_to_index:
    vector = model[word]
    print(f"Vector for '{word}': {vector[:10]}... (first 10 dimensions)")
    print(f"Vector dimension: {len(vector)}")
else:
    print(f"'{word}' not found in vocabulary.")
```

```
Vocabulary size: 400000 words
Vector for 'computer': [ 0.079084 -0.81504   1.7901    0.91653   0.10797  -0.55628  -0.84427
 -1.4951    0.13418   0.63627 ]... (first 10 dimensions)
Vector dimension: 50
```

```
word_list = [
```

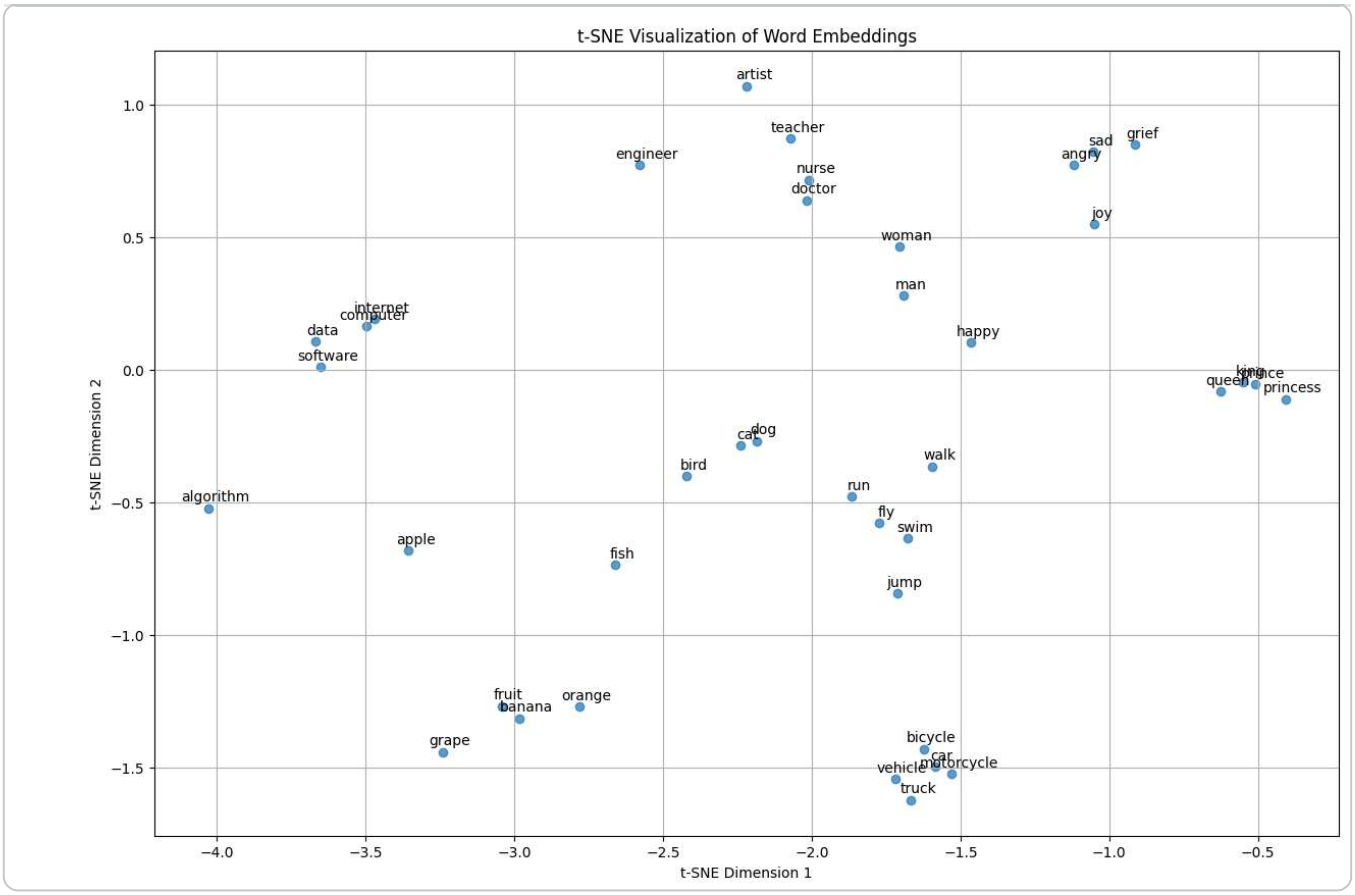
```
word_list = [
    'king', 'queen', 'man', 'woman', 'prince', 'princess',
    'apple', 'banana', 'orange', 'grape', 'fruit',
    'car', 'truck', 'bicycle', 'motorcycle', 'vehicle',
    'doctor', 'nurse', 'engineer', 'teacher', 'artist',
    'happy', 'sad', 'angry', 'joy', 'grief',
    'run', 'walk', 'jump', 'swim', 'fly',
    'computer', 'software', 'internet', 'data', 'algorithm',
    'cat', 'dog', 'bird', 'fish'
]

selected_words = []
word_vectors = []

for word in word_list:
    if word in model.key_to_index:
        selected_words.append(word)
        word_vectors.append(model[word])
    else:
        print(f"Warning: '{word}' not found in model vocabulary and will be skipped.")

word_vectors_array = np.array(word_vectors)

print(f"Successfully extracted vectors for {len(selected_words)} out of {len(word_list)} chosen wo
print(f"Shape of word_vectors_array: {word_vectors_array.shape}")
```

```
Successfully extracted vectors for 40 out of 40 chosen words.
Shape of word_vectors_array: (40, 50)
```

```
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(word_vectors_array)

tsne_df = pd.DataFrame(data=tsne_results, columns=['x', 'y'])
tsne_df['word'] = selected_words

print("First 5 rows of the t-SNE DataFrame:")
print(tsne_df.head())
print(f"Shape of the t-SNE DataFrame: {tsne_df.shape}")
```

```
First 5 rows of the t-SNE DataFrame:
          x         y    word
0 -0.548899 -0.047648    king
1 -0.625454 -0.081488   queen
2 -1.690122  0.279161     man
3 -1.704404  0.463638   woman
4 -0.507809 -0.054847  prince
Shape of the t-SNE DataFrame: (40, 3)
```

```
plt.figure(figsize=(15, 10))
plt.scatter(tsne_df['x'], tsne_df['y'], alpha=0.7)

for i, row in tsne_df.iterrows():
    plt.annotate(row['word'], (row['x'], row['y']), textcoords="offset points", xytext=(5, 5), ha=

plt.title('t-SNE Visualization of Word Embeddings')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.grid(True)
plt.show()
```


t-SNE Visualization of Word Embeddings
t-SNE Dimension 1
t-SNE Dimension 2
artist
teacher
engineer
nurse
doctor
sad
grief
angry
joy
woman
man
happy
internet
computer
data
software
king
queen
prince
princess
cat
dog
bird
walk
run
fly
swim
algorithm
apple
fish
jump
fruit
banana
orange
grape
bicycle
car
motorcycle
vehicle
truck