```
# Install the gensim library, which is used for working with word embeddings.
!pip install gensim

# gensim - to load and work with pre-trained word embedding models
import gensim.downloader as api

# numpy - for numerical operations on vectors
import numpy as np

# pandas - to organize similarity results in table format
import pandas as pd

# matplotlib - for visualization
import matplotlib.pyplot as plt

# sklearn - for PCA dimensionality reduction, used for visualizing high-dimensional data in 2D
from sklearn.decomposition import PCA

# nltk - for optional NLP utilities, though not explicitly used in the subsequent code, it's a common NLP library.
import nltk

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

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.4.0)
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)
Libraries imported successfully.
```

```
print("Loading Word2Vec model...")
# Load a pre-trained Word2Vec model from Google News, which has 300-dimensional vectors.
model = api.load("word2vec-google-news-300")  # 300-dimensional vectors

print("Model loaded successfully!")

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

# Display an example word vector for the word "king".
word = "king"
print(f"\nVector for '{word}':\n")
print(model[word])

# Print the length (dimensionality) of the word vector.
print("\nVector Length:", len(model[word]))
```

```
Loading Word2Vec model...
Model loaded successfully!
Vocabulary Size: 3000000

Vector for 'king':

[ 1.25976562e-01  2.97851562e-02  8.60595703e-03  1.39648438e-01
 -2.56347656e-02 -3.61328125e-02  1.11816406e-01 -1.98242188e-01
  5.12695312e-02  3.63281250e-01 -2.42187500e-01 -3.02734375e-01
 -1.77734375e-01 -2.49023438e-02 -1.67968750e-01 -1.69921875e-01
  3.46679688e-02  5.21850586e-03  4.63867188e-02  1.28906250e-01
  1.36718750e-01  1.12792969e-01  5.95703125e-02  1.36718750e-01
  1.01074219e-01 -1.76757812e-01 -2.51953125e-01  5.98144531e-02
  3.41796875e-01 -3.11279297e-02  1.04492188e-01  6.17675781e-02
  1.24511719e-01  4.00390625e-01 -3.22265625e-01  8.39843750e-02
  3.90625000e-02  5.85937500e-03  7.03125000e-02  1.72851562e-01
  1.38671875e-01 -2.31445312e-01  2.83203125e-01  1.42578125e-01
  3.41796875e-01 -2.39257812e-02 -1.09863281e-01  3.32031250e-02
 -5.46875000e-02  1.53198242e-02 -1.62109375e-01  1.58203125e-01
 -2.59765625e-01  2.01416016e-02 -1.63085938e-01  1.35803223e-03
 -1.44531250e-01 -5.68847656e-02  4.29687500e-02 -2.46582031e-02
  1.85546875e-01  4.47265625e-01  9.58251953e-03  1.31835938e-01
  9.86328125e-02 -1.85546875e-01 -1.00097656e-01 -1.33789062e-01
 -1.25000000e-01  2.83203125e-01  1.23046875e-01  5.32226562e-02
 -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
 -1.64062500e-01  1.15356445e-02  4.45312500e-01 -2.15820312e-01
 -1.11328125e-01 -1.92382812e-01  1.70898438e-01 -1.25000000e-01
  2.65502930e-03  1.92382812e-01 -1.74804688e-01  1.39648438e-01
  2.92968750e-01  1.13281250e-01  5.95703125e-02 -6.39648438e-02
  9.96093750e-02 -2.72216797e-02  1.96533203e-02  4.27246094e-02
 -2.46093750e-01  6.39648438e-02 -2.25585938e-01 -1.68945312e-01
  2.89916992e-03  8.20312500e-02  3.41796875e-01  4.32128906e-02
  1.32812500e-01  1.42578125e-01  7.61718750e-02  5.98144531e-02
 -1.19140625e-01  2.74658203e-03 -6.29882812e-02 -2.72216797e-02
 -4.82177734e-03 -8.20312500e-02 -2.49023438e-02 -4.00390625e-01
 -1.06933594e-01  4.24804688e-02  7.76367188e-02 -1.16699219e-01
```

```
# gensim - to load and work with pre-trained word embedding models
import gensim.downloader as api

# Load a pre-trained Word2Vec model from Google News, which has 300-dimensional vectors.
model = api.load("word2vec-google-news-300")  # 300-dimensional vectors

# Define a list of word pairs for which to calculate cosine similarity.
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("man", "woman"),
    ("computer", "laptop"),
    ("school", "college"),
    ("river", "water"),
    ("food", "apple"),
    ("teacher", "student")
]

# Initialize an empty list to store similarity results.
similarities = []

# Iterate through each word pair, calculate their cosine similarity, and store the result.
for w1, w2 in word_pairs:
    sim = model.similarity(w1, w2)
    similarities.append((w1, w2, sim))

# Create a pandas DataFrame from the similarity results for easy viewing.
df = pd.DataFrame(similarities, columns=["Word 1", "Word 2", "Cosine Similarity"])
# Display the DataFrame.
df
```

```
ERROR:root:Internal Python error in the inspect module.
Below is the traceback from this internal error.

Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/IPython/core/interactiveshell.py", line 3553, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "/tmp/ipython-input-4069261106.py", line 5, in <cell line: 0>
    model = api.load("word2vec-google-news-300")  # 300-dimensional vectors
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gensim/downloader.py", line 503, in load
    return module.load_data()
           ^^^^^^^^^^^^^^^^^^
  File "/root/gensim-data/word2vec-google-news-300/__init__.py", line 8, in load_data
    model = KeyedVectors.load_word2vec_format(path, binary=True)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gensim/models/keyedvectors.py", line 1721, in load_word2vec_format
    return _load_word2vec_format(
           ^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/gensim/models/keyedvectors.py", line 2067, in _load_word2vec_format
    _word2vec_read_binary(
  File "/usr/local/lib/python3.12/dist-packages/gensim/models/keyedvectors.py", line 1960, in _word2vec_read_binary
    new_chunk = fin.read(binary_chunk_size)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/gzip.py", line 338, in read
    return self._buffer.read(size)
           ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/_compression.py", line 68, in readinto
    data = self.read(len(byte_view))
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/gzip.py", line 554, in read
    uncompress = self._decompressor.decompress(b"", size)
                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/IPython/core/interactiveshell.py", line 2099, in showtraceback
    stb = value._render_traceback_()
          ^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'KeyboardInterrupt' object has no attribute '_render_traceback_'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/IPython/core/ultratb.py", line 1101, in get_records
    return _fixed_getinnerframes(etb, number_of_lines_of_context, tb_offset)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/IPython/core/ultratb.py", line 248, in wrapped
    return f(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/IPython/core/ultratb.py", line 281, in _fixed_getinnerframes
    records = fix_frame_records_filenames(inspect.getinnerframes(etb, context))
                                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/inspect.py", line 1769, in getinnerframes
    traceback_info = getframeinfo(tb, context)
                     ^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
# gensim - to load and work with pre-trained word embedding models
import gensim.downloader as api

# Load a pre-trained Word2Vec model from Google News, which has 300-dimensional vectors.
model = api.load("word2vec-google-news-300")  # 300-dimensional vectors

# Define a list of words for which to find the most similar terms.
test_words = ["king", "university", "doctor", "india", "computer"]

# Iterate through each test word.
for word in test_words:
    print(f"\nTop 5 words similar to '{word}':")
    # Find the top 5 most similar words using the pre-trained model.
    similar_words = model.most_similar(word, topn=5)
    # Print each similar word along with its similarity score.
    for w, score in similar_words:
        print(f"{w} ({score:.3f})")
```

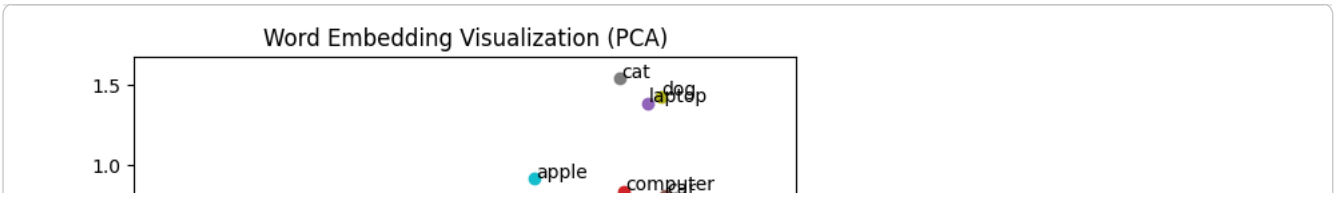
```
           ^^^^^^^^^^
KeyboardInterrupt
Top 5 words similar to 'king':
---------------------------------------------------------------------------
kings (0.714)
KeyboardInterrupt                         Traceback (most recent call last)
queen (0.651)
    [... skipping hidden 1 frame]
monarch (0.641)
crown prince (0.620)
/tmp/ipython-input-4069261106.py in <cell line: 0>()
prince (0.616)
      4 # Load a pre-trained Word2Vec model from Google News, which has 300-dimensional vectors.
Top 5 words similar to 'university':
----> 5 model = api.load("word2vec-google-news-300")  # 300-dimensional vectors
      6
universities (0.700)
faculty (0.678)
                                14 frames
unversity (0.676)
KeyboardInterrupt:
```

```
undergraduate (0.659)
university (0.659)

Top 5 words similar to 'doctor':
physician (0.781)
doctors (0.748)
gynecologist (0.695)
surgeon (0.679)
dentist (0.679)
Top 5 words similar to 'india':
indian (0.697)
usa (0.684)
pakistan (0.682)
chennai (0.668)
america (0.659)

Top 5 words similar to 'computer':
computers (0.798)
laptop (0.664)
laptop_computer (0.655)
Computer (0.647)
com_puter (0.608)
```

```
print("Analogy 1: king - man + woman = ?")
# Calculate the analogy: king - man + woman, expecting 'queen'.
print(model.most_similar(positive=["king", "woman"], negative=["man"], topn=1))

print("\nAnalogy 2: paris - france + india = ?")
# Calculate the analogy: paris - france + india, expecting an Indian city.
print(model.most_similar(positive=["paris", "india"], negative=["france"], topn=1))

print("\nAnalogy 3: teacher - school + hospital = ?")
# Calculate the analogy: teacher - school + hospital, expecting a profession in a hospital.
print(model.most_similar(positive=["teacher", "hospital"], negative=["school"], topn=1))
```

```
# gensim - to load and work with pre-trained word embedding models
import gensim.downloader as api

# Load a pre-trained Word2Vec model from Google News, which has 300-dimensional vectors.
model = api.load("word2vec-google-news-300")  # 300-dimensional vectors

# sklearn - for PCA dimensionality reduction, used for visualizing high-dimensional data in 2D
from sklearn.decomposition import PCA

# matplotlib - for visualization
import matplotlib.pyplot as plt

# Define a list of words to be visualized.
words = ["king", "queen", "man", "woman",
         "doctor", "nurse", "hospital", "teacher",
         "school", "india", "china", "paris",
         "france", "computer", "laptop", "car",
         "bus", "cat", "dog", "apple"]

# Get the vector representation for each word in the list.
victors = [model[word] for word in words]

# Reduce the 300-dimensional word vectors to 2 dimensions using PCA (Principal Component Analysis).
pca = PCA(n_components=2)
reduced = pca.fit_transform(victors)

# Plot the 2D reduced word embeddings.
plt.figure()
for i, word in enumerate(words):
    # Plot each word as a scatter point.
    plt.scatter(reduced[i, 0], reduced[i, 1])
    # Add the word label next to its scatter point.
    plt.text(reduced[i, 0]+0.01, reduced[i, 1]+0.01, word)

# Add a title to the plot.
plt.title("Word Embedding Visualization (PCA)")
# Display the plot.
plt.show()
```

```
During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
AttributeError: 'KeyboardInterrupt' object has no attribute '_render_traceback_'
During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
    [... skipping hidden 1 frame]
/usr/local/lib/python3.12/dist-packages/IPython/core/ultratb.py in find_recursion(etype, value, records)
    380     # first frame (from in to out) that looks different.
    381     if not is_recursion_error(etype, value, records):
--> 382         return len(records), 0
    383
    384     # Select filename, lineno, func_name to track frames with
TypeError: object of type 'NoneType' has no len()
```


Word Embedding Visualization (PCA)
1.5
1.0
cat
dog
laptop
apple
computer