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

# Used to load pre-trained Word2Vec / GloVe models
import gensim
import gensim.downloader as api

# Used for numerical operations (vectors, cosine similarity)
import numpy as np

# Used for data handling (optional for tables)
import pandas as pd

# Used for tokenization or text utilities (if needed)
import nltk

# Used for visualization
import matplotlib.pyplot as plt

# Used for dimensionality reduction (PCA)
from sklearn.decomposition import PCA
```

```
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 62.5 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

```
# Load pre-trained Word2Vec model (100-dimensional)
model = api.load("glove-wiki-gigaword-100")

print("Vocabulary Size:", len(model.key_to_index))

# Display vector for a word
word = "king"
vector = model[word]

print("Vector for 'king':")
print(vector)
print("Vector length:", len(vector))
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
Vocabulary Size: 400000
Vector for 'king':
[-0.32307  -0.87616   0.21977   0.25268   0.22976   0.7388   -0.37954
 -0.35307  -0.84369  -1.1113   -0.30266   0.33178  -0.25113   0.30448
 -0.077491 -0.89815   0.092496 -1.1407   -0.58324   0.66869  -0.23122
 -0.95855   0.28262  -0.078848  0.75315   0.26584   0.3422   -0.33949
  0.95608   0.065641  0.45747   0.39835   0.57965   0.39267  -0.21851
  0.58795  -0.55999   0.63368  -0.043983 -0.68731  -0.37841   0.38026
  0.61641  -0.88269  -0.12346  -0.37928  -0.38318   0.23868   0.6685
 -0.43321  -0.11065   0.081723  1.1569    0.78958  -0.21223  -2.3211
 -0.67806   0.44561   0.65707   0.1045    0.46217   0.19912   0.25802
  0.057194  0.53443  -0.43133  -0.34311   0.59789  -0.58417   0.068995
  0.23944  -0.85181   0.30379  -0.34177  -0.25746  -0.031101 -0.16285
  0.45169  -0.91627   0.64521   0.73281  -0.22752   0.30226   0.044801
 -0.83741   0.55006  -0.52506  -1.7357    0.4751   -0.70487   0.056939
 -0.7132    0.089623  0.41394  -1.3363   -0.61915  -0.33089  -0.52881
  0.16483  -0.98878 ]
Vector length: 100
```

```
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("man", "woman"),
    ("paris", "france"),
    ("apple", "fruit"),
    ("teacher", "student"),
```

```
    ("river", "water"),
    ("computer", "keyboard")
]

for w1, w2 in word_pairs:
    similarity = model.similarity(w1, w2)
    print(f"Similarity between {w1} and {w2}: {similarity:.4f}")
```

```
Similarity between doctor and nurse: 0.7522
Similarity between cat and dog: 0.8798
Similarity between car and bus: 0.7373
Similarity between king and queen: 0.7508
Similarity between man and woman: 0.8323
Similarity between paris and france: 0.7482
Similarity between apple and fruit: 0.5359
Similarity between teacher and student: 0.8083
Similarity between river and water: 0.6306
Similarity between computer and keyboard: 0.5418
```

```
words_to_check = ["king", "university", "india", "computer", "music"]

for word in words_to_check:
    print(f"\nTop 5 words similar to '{word}':")
    for similar_word, score in model.most_similar(word, topn=5):
        print(f"{similar_word} ({score:.4f})")
```

```

Top 5 words similar to 'king':
prince (0.7682)
queen (0.7508)
son (0.7021)
brother (0.6986)
monarch (0.6978)

Top 5 words similar to 'university':
college (0.8294)
harvard (0.8156)
yale (0.8114)
professor (0.8104)
graduate (0.7993)

Top 5 words similar to 'india':
pakistan (0.8370)
indian (0.7802)
delhi (0.7712)
bangladesh (0.7662)
lanka (0.7639)

Top 5 words similar to 'computer':
computers (0.8752)
software (0.8373)
technology (0.7642)
pc (0.7366)
hardware (0.7290)

Top 5 words similar to 'music':
musical (0.8128)
songs (0.7978)
dance (0.7897)
pop (0.7863)
recording (0.7651)
```

```
# king - man + woman
result1 = model.most_similar(positive=["king", "woman"], negative=["man"], topn=1)
print("king - man + woman =", result1)

# paris - france + india
result2 = model.most_similar(positive=["paris", "india"], negative=["france"], topn=1)
print("paris - france + india =", result2)

# teacher - school + hospital
result3 = model.most_similar(positive=["teacher", "hospital"], negative=["school"], topn=1)
print("teacher - school + hospital =", result3)
```

```
king - man + woman = [('queen', 0.7698540687561035)]
paris - france + india = [('delhi', 0.8654932975769043)]
teacher - school + hospital = [('nurse', 0.7798740267753601)]
```

```
words = ["king", "queen", "man", "woman",
         "paris", "france", "india", "delhi",
         "doctor", "nurse", "teacher", "student",
         "cat", "dog", "lion", "tiger"]

vectors = np.array([model[word] for word in words])

# Reduce to 2D
pca = PCA(n_components=2)
reduced = pca.fit_transform(vectors)

plt.figure(figsize=(10,8))
plt.scatter(reduced[:,0], reduced[:,1])

for i, word in enumerate(words):
    plt.annotate(word, (reduced[i,0], reduced[i,1]))

plt.title("Word Embedding Visualization (PCA)")
plt.show()
```