STEP 1 — Import Required Libraries

```
!pip install gensim
# gensim is used to load and work with pre-trained Word2Vec and GloVe models
from gensim.models import KeyedVectors

# numpy is used for vector calculations
import numpy as np

# pandas is used to store similarity results in tabular form
import pandas as pd

# matplotlib is used to visualize word vectors
import matplotlib.pyplot as plt

# PCA is used to reduce high-dimensional vectors to 2D for plotting
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
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 72.5 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

STEP 2 — Load Pre-trained Word2Vec Model

```
# Path to the downloaded Word2Vec file (no longer needed for gensim.downloader)
# word2vec_path = "GoogleNews-vectors-negative300.bin.gz"

# Using gensim.downloader to download and load the Word2Vec model
import gensim.downloader as api

# Download and load the 'word2vec-google-news-300' model
# This model is approximately 1.5 GB and might take a few minutes to download.
w2v_model = api.load("word2vec-google-news-300")

# Print total number of words in vocabulary
print("Vocabulary size:", len(w2v_model))

# Display first 10 values of the vector for a sample word
print("Vector for 'king':")
print(w2v_model["king"][:10])
```

```
[==================================================] 100.0% 1662.8/1662.8MB downloaded
Vocabulary size: 3000000
Vector for 'king':
[ 0.12597656  0.02978516  0.00860596  0.13964844 -0.02563477 -0.03613281
  0.11181641 -0.19824219  0.05126953  0.36328125]
```

STEP 3 — Word Similarity (Cosine Similarity)

```
# List of word pairs to compare similarity
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("apple", "banana"),
    ("teacher", "student"),
    ("man", "woman"),
    ("paris", "france"),
    ("coffee", "tea"),
    ("computer", "laptop")
]

# Store similarity results
results = []

for w1, w2 in word_pairs:
    similarity = w2v_model.similarity(w1, w2)
    results.append([w1, w2, similarity])

# Convert results into DataFrame
similarity_df = pd.DataFrame(
    results,
    columns=["Word 1", "Word 2", "Cosine Similarity"]
)

similarity_df
```

	Word 1	Word 2	Cosine Similarity
0	doctor	nurse	0.631952
1	cat	dog	0.760946
2	car	bus	0.469337
3	king	queen	0.651096
4	apple	banana	0.531841
5	teacher	student	0.630137
6	man	woman	0.766401
7	paris	france	0.555080
8	coffee	tea	0.563529
9	computer	laptop	0.664049

Next steps: Generate code with similarity_df New interactive sheet

STEP 4 — Nearest Neighbor Exploration

```
# Words for nearest neighbor analysis
words = ["king", "university", "doctor", "computer", "india"]

for word in words:
    print(f"\nTop 5 similar words to '{word}':")
    for similar_word, score in w2v_model.most_similar(word, topn=5):
        print(similar_word, ":", score)
```

```
Top 5 similar words to 'king':
kings : 0.7138045430183411
queen : 0.6510956883430481
monarch : 0.6413194537162781
crown_prince : 0.6204220056533813
prince : 0.6159993410110474

Top 5 similar words to 'university':
universities : 0.7003918886184692
faculty : 0.6780907511711121
unversity : 0.6758289933204651
undergraduate : 0.6587094664573669
univeristy : 0.6585438251495361

Top 5 similar words to 'doctor':
physician : 0.7806021571159363
doctors : 0.747657299041748
gynecologist : 0.6947518587112427
surgeon : 0.6793398261070251
dentist : 0.6785441040992737

Top 5 similar words to 'computer':
computers : 0.7979379892349243
laptop : 0.6640493273735046
laptop_computer : 0.6548868417739868
Computer : 0.647333562374115
com_puter : 0.6082080006599426

Top 5 similar words to 'india':
indian : 0.6967039704322815
usa : 0.6836211085319519
pakistan : 0.681516706943512
chennai : 0.6675503253936768
america : 0.6589399576187134
```

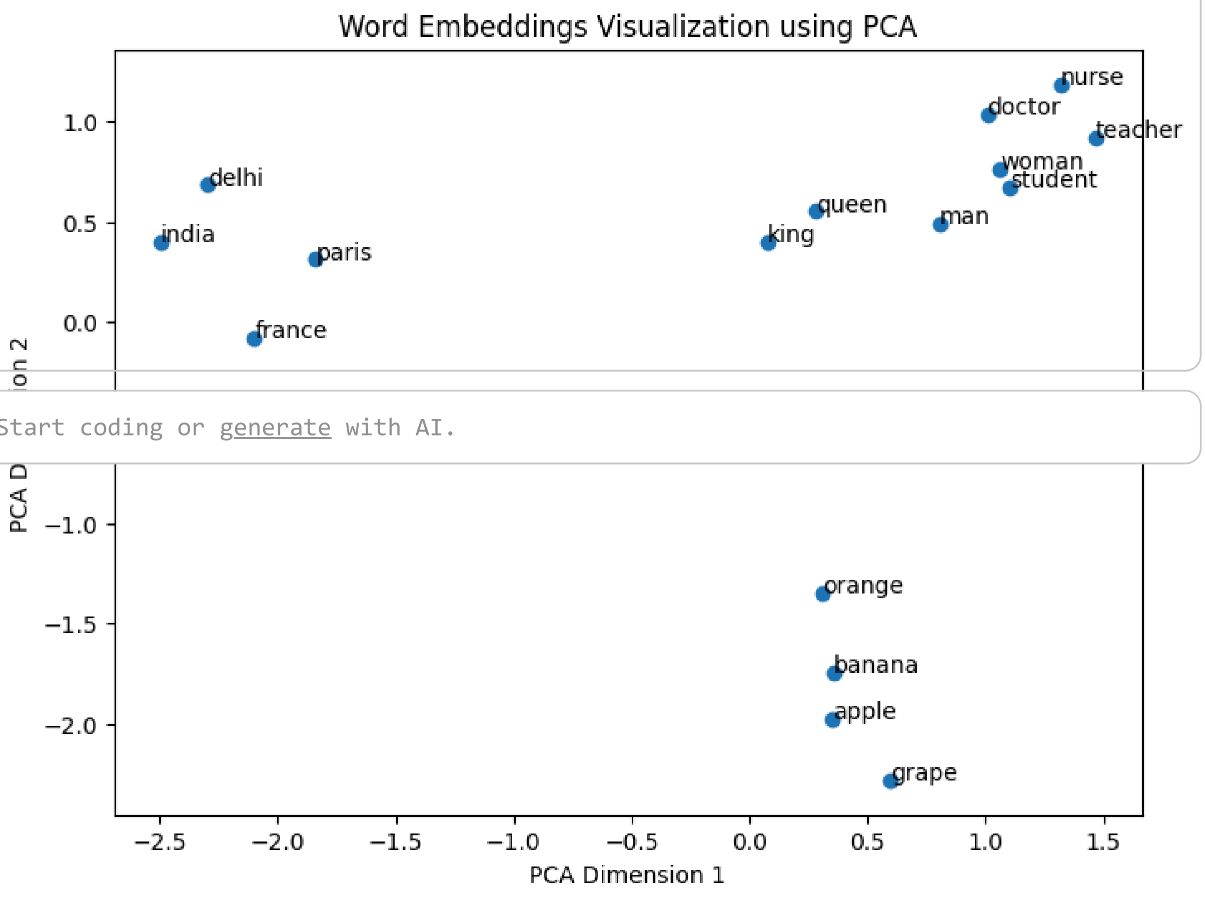
STEP 5 — Word Analogy Tasks

```
# king - man + woman
print("king - man + woman =")
print(w2v_model.most_similar(
    positive=["king", "woman"],
    negative=["man"]
))

# paris - france + india
print("\nparis - france + india =")
print(w2v_model.most_similar(
    positive=["paris", "india"],
    negative=["france"]
))

# teacher - school + hospital
print("\nteacher - school + hospital =")
print(w2v_model.most_similar(
    positive=["teacher", "hospital"],
    negative=["school"]
))
```

```
king - man + woman =
[('queen', 0.7118193507194519), ('monarch', 0.6189674139022827), ('princess', 0.590243

paris - france + india =
[('chennai', 0.5442505478858948), ('delhi', 0.5149926543235779), ('mumbai', 0.50243413

teacher - school + hospital =
[('Hospital', 0.6331106424331665), ('nurse', 0.6280134320259094), ('hopsital', 0.62173
```

STEP 6 — Visualization Using PCA (Optional)

```
# Words selected for visualization
viz_words = [
    "king", "queen", "man", "woman",
    "paris", "france", "india", "delhi",
    "doctor", "nurse", "teacher", "student",
    "apple", "banana", "orange", "grape"
]

# Get word vectors
vectors = np.array([w2v_model[word] for word in viz_words])

# Reduce dimensions from 300D to 2D
pca = PCA(n_components=2)
reduced_vectors = pca.fit_transform(vectors)

# Plot the vectors
plt.figure(figsize=(8, 6))
plt.scatter(reduced_vectors[:, 0], reduced_vectors[:, 1])

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

plt.title("Word Embeddings Visualization using PCA")
plt.xlabel("PCA Dimension 1")
plt.ylabel("PCA Dimension 2")
plt.show()
```



Start coding or generate with AI.