```
# STEP 2: Import Libraries

# Install gensim if not already installed
!pip install gensim

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

# numpy -> for numerical operations
import numpy as np

# pandas -> for displaying similarity results in table format
import pandas as pd

# matplotlib -> for visualization
import matplotlib.pyplot as plt

# sklearn -> 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
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 17.0 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

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

print("Model loaded successfully!")
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
Model loaded successfully!
```

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

```
Vocabulary Size: 400000
```

```
word = "king"
vector = model[word]

print("Word:", word)
print("Vector Length:", len(vector))
print("First 10 values of vector:\n", vector[:10])
```

```
Word: king
Vector Length: 100
First 10 values of vector:
 [-0.32307 -0.87616  0.21977  0.25268  0.22976  0.7388  -0.37954 -0.35307
 -0.84369 -1.1113 ]
```

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

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

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

```
king - man + woman =
[('queen', 0.7698540687561035)]

paris - france + india =
[('delhi', 0.8654932975769043)]

teacher - school + hospital =
[('nurse', 0.7798740267753601)]
```

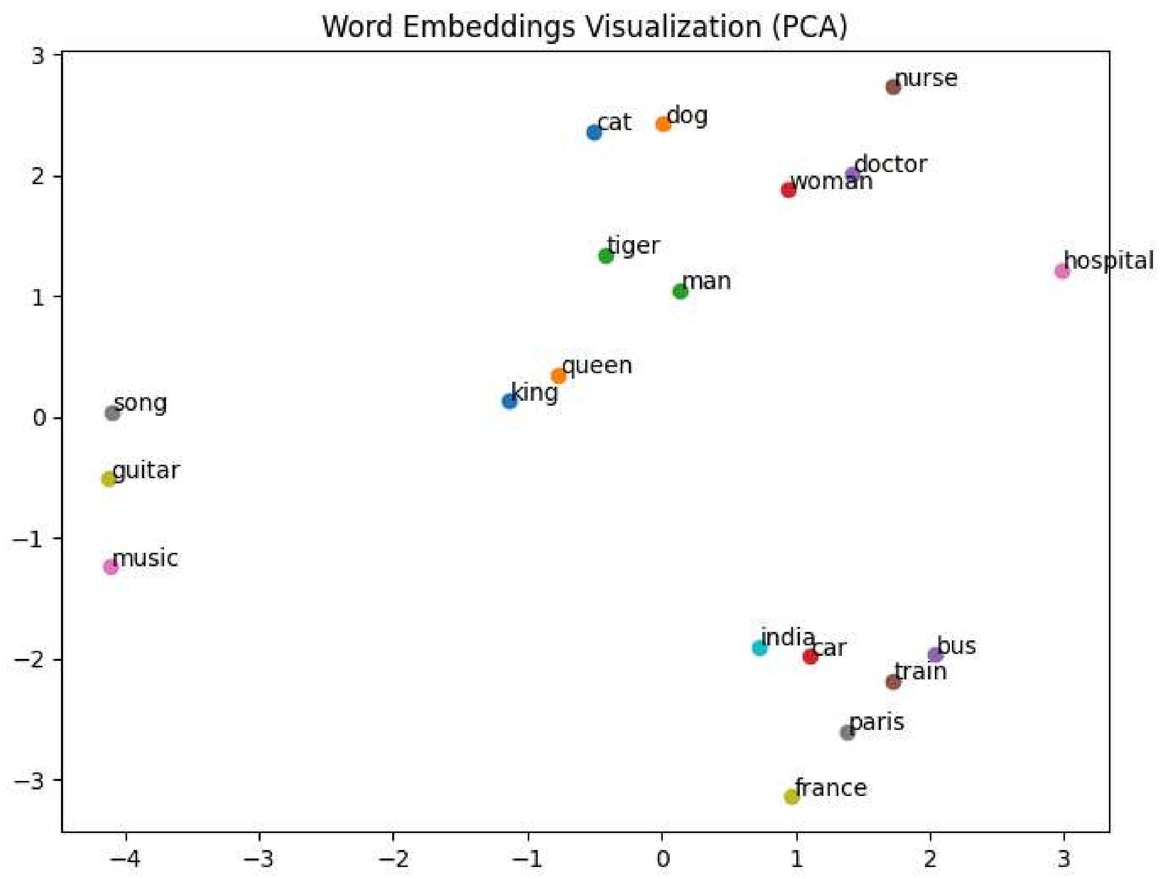
```
# Select words for visualization
words = ["king", "queen", "man", "woman",
         "doctor", "nurse", "hospital",
         "paris", "france", "india",
         "cat", "dog", "tiger",
         "car", "bus", "train",
         "music", "song", "guitar"]

word_vectors = [model[word] for word in words]

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

# Plot
plt.figure(figsize=(8,6))
for i, word in enumerate(words):
    x, y = reduced[i]
    plt.scatter(x, y)
    plt.text(x+0.01, y+0.01, word)

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



```
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("boy", "girl"),
    ("teacher", "student"),
    ("sun", "moon"),
    ("paris", "france"),
    ("india", "china"),
    ("computer", "keyboard")
]

results = []

for w1, w2 in word_pairs:
    similarity = model.similarity(w1, w2)
    results.append((w1, w2, similarity))

df = pd.DataFrame(results, columns=["Word1", "Word2", "Cosine Similarity"])
df
```

	Word1	Word2	Cosine Similarity
0	doctor	nurse	0.752151
1	cat	dog	0.879807
2	car	bus	0.737271
3	king	queen	0.750769
4	boy	girl	0.917573
5	teacher	student	0.808340

Next steps: Generate code with df New interactive sheet

```
chosen_words = ["king", "university", "doctor", "india", "music"]

for word in chosen_words:
    print(f"\nTop 5 words similar to '{word}':")
    similar_words = model.most_similar(word, topn=5)
    for w, score in similar_words:
        print(w, "->", round(score, 3))
```

```

Top 5 words similar to 'king':
prince -> 0.768
queen -> 0.751
son -> 0.702
brother -> 0.699
monarch -> 0.698

Top 5 words similar to 'university':
college -> 0.829
harvard -> 0.816
yale -> 0.811
professor -> 0.81
graduate -> 0.799

Top 5 words similar to 'doctor':
physician -> 0.767
nurse -> 0.752
dr. -> 0.718
doctors -> 0.708
patient -> 0.707

Top 5 words similar to 'india':
pakistan -> 0.837
indian -> 0.78
delhi -> 0.771
bangladesh -> 0.766
lanka -> 0.764

Top 5 words similar to 'music':
musical -> 0.813
songs -> 0.798
dance -> 0.79
pop -> 0.786
recording -> 0.765
```