```
!pip install gensim
import gensim
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity
```

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

```
# 1️⃣ Remove corrupted files
!rm -f GoogleNews-vectors-negative300.bin
!rm -f GoogleNews-vectors-negative300.bin.gz

# 2️⃣ Download again (this file is ~1.5GB — wait fully)
!wget -c https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz

# 3️⃣ Unzip
!gunzip GoogleNews-vectors-negative300.bin.gz
```

```
--2026-02-11 05:50:54--  https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 54.231.226.0, 16.182.101.216, 54.231.196.152, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|54.231.226.0|:443... connected.
HTTP request sent, awaiting response... 404 Not Found
2026-02-11 05:50:55 ERROR 404: Not Found.

gzip: GoogleNews-vectors-negative300.bin.gz: No such file or directory
```

```
import gensim.downloader as api
model = api.load("glove-wiki-gigaword-100")
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
```

```
[0.125, -0.345, 0.889, ...]
```

```
[0.125, -0.345, 0.889, Ellipsis]
```

```
pairs = [
    ('doctor', 'nurse'),
    ('cat', 'dog'),
    ('car', 'bus'),
    ('king', 'queen'),
    ('boy', 'girl'),
    ('school', 'university'),
    ('cpu', 'computer'),
    ('ai', 'machine'),
    ('teacher', 'student'),
    ('india', 'china')
]

for w1, w2 in pairs:
    print(f"{w1} - {w2} : ", model.similarity(w1, w2))
```

```
doctor - nurse :  0.7521509
cat - dog :  0.8798075
car - bus :  0.7372708
king - queen :  0.7507691
boy - girl :  0.91757303
school - university :  0.7548452
cpu - computer :  0.41330045
ai - machine :  0.37361014
teacher - student :  0.80833995
india - china :  0.5997108
```

```
words = ['king', 'university', 'computer', 'india', 'doctor']
```

```
for word in words:
    print(f"\nTop similar words for {word}:")
    print(model.most_similar(word, topn=5))
```

```
Top similar words for king:
[('prince', 0.7682328820228577), ('queen', 0.7507690787315369), ('son', 0.7020888328552246), ('brother', 0.6985775232315063), ('

Top similar words for university:
[('college', 0.8294212818145752), ('harvard', 0.8156033754348755), ('yale', 0.8113803267478943), ('professor', 0.810378491878509

Top similar words for computer:
[('computers', 0.8751984238624573), ('software', 0.8373122215270996), ('technology', 0.7642159461975098), ('pc', 0.7366448640823

Top similar words for india:
[('pakistan', 0.8370323777198792), ('indian', 0.780203104019165), ('delhi', 0.7712194323539734), ('bangladesh', 0.76616412401199

Top similar words for doctor:
[('physician', 0.7673240303993225), ('nurse', 0.75215083360672), ('dr.', 0.7175194025039673), ('doctors', 0.7080884575843811), (
```

```
print(model.most_similar(positive=['king', 'woman'], negative=['man'], topn=1))
print(model.most_similar(positive=['paris', 'india'], negative=['france'], topn=1))
print(model.most_similar(positive=['teacher', 'hospital'], negative=['school'], topn=1))
```

```
[('queen', 0.7698540687561035)]
[('delhi', 0.8654932975769043)]
[('nurse', 0.7798740267753601)]
```

```
selected_words = ['king','queen','man','woman','paris','france','india',
                  'doctor','nurse','teacher','school','hospital']

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

pca = PCA(n_components=2)
result = pca.fit_transform(vectors)

plt.figure(figsize=(8,6))
plt.scatter(result[:,0], result[:,1])

for i, word in enumerate(selected_words):
    plt.annotate(word, xy=(result[i,0], result[i,1]))

plt.show()
```